Virtual Video Camera: a System for Free Viewpoint Video of Arbitrary Dynamic Scenes

Von der Carl-Friedrich-Gauß Fakultät

Technische Universität Carola-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines

Doktoringenieurs (Dr.-Ing.)

genehmigte

Dissertation

von	Christian Lipski
geboren in	Braunschweig
am	12. Mai 1981

Eingereicht am: 27.03.2013

Disputation am: 05.06.2013

1. Referent: Prof. Dr.-Ing. Marcus Magnor

2. Referent: Prof. Adrian Hilton, DPhil., BSc.(hons.), CEng

(2013)

Abstract

The *Virtual Video Camera* project strives to create free viewpoint video from "casually" captured multi-view data. Multiple video streams of a dynamic scene are captured with off-the-shelf camcorders, and the user can re-render the scene from novel perspectives. In this thesis the algorithmic core of the *Virtual Video Camera* is presented. This includes the algorithm for image correspondence estimation as well as the image-based renderer. Furthermore, its application in the context of an actual video production is showcased, and the rendering and image processing pipeline is extended to incorporate depth information.

Kurzfassung

Das *Virtual Video Camera* Projekt dient der Erzeugung von *Free Viewpoint Video* Ansichten von *Multi-View* Aufnahmen: Material mehrerer Videoströme wird hierzu mit handelsüblichen Camcordern aufgezeichnet. Im Anschluss kann die Szene aus beliebigen, von den ursprünglichen Kameras nicht abgedeckten Blickwinkeln betrachtet werden. In dieser Dissertation wird der algorithmische Kern der *Virtual Video Camera* vorgestellt. Dies beinhaltet das Verfahren zur Bildkorrespondenzschätzung sowie den bildbasierten Renderer. Darüber hinaus wird die Anwendung im Kontext einer Videoproduktion beleuchtet. Dazu wird die bildbasierte Erzeugung neuer Blickpunkte um die Erzeugung und Einbindung von Tiefeninformationen erweitert.

Summary

The first decade of the 21^{st} century has seen a tremendous increase in the creation of digital video. Recording and processing hardware has reached maturity and generally low price levels. While in the past, the recording and distribution of high-resolution digital media had been restricted to the professional movie industry, it is now available to a broad public. Independent film makers and hobbyists can, potentially, produce high quality content on a shoestring budget. Nevertheless, when it comes to producing stunning visual effects for movies, a lot of effort and planning is required. Often, studio setups with specialized capturing hardware and controllable lighting have to be used. Cameras, actors, backgrounds and props are cluttered with markers to track and analyze scene dynamics. To realize some advanced visual effects, it is not uncommon to build elaborate 3D models of the scene and the actors.

Goal of this thesis is to provide solutions for visual effects production on a small budget that require little manual work. The video processing and rendering algorithms presented in this thesis allow film makers to create *free viewpoint videos* of a scene: They can re-render the scene from novel viewpoints that have not been observed by an actual camera during the shooting of the video. This is made possible by capturing the scene with a handful of consumer-grade camcorders and synthesizing novel views. By using image-based techniques for rendering, the system circumvents the necessity of reconstructing hard-to-compute, error-prone geometric models of the scene. This makes it possible to capture material with low-cost equipment and without additional on-set hassle with calibration patterns, laser scanners, additional lighting, or synchronization hardware. Even outdoor recordings are possible with little set-up time.

In the first part of this thesis, the pairwise image correspondence estimation needed for view interpolation is documented. In the second part, the image warping- and blending-based renderer is presented. To prove the practicability of the *Virtual Video Camera*, its application in the production of the award-winning, stereoscopic free viewpoint music video "Who Cares" by Symbiz Sound are showcased. Based on the lessons learned in the creation of the "Who Cares" music video, the renderer is extended to a hybrid between correspondence and depth-image-based rendering in the third part of the thesis. By jointly considering depth and correspondence information, processing robustness is increased and new applications and effects are made possible, such as view extrapolation, image stabilization, panorama creation or relighting.

Zusammenfassung

Die erste Dekade des 21. Jahrhunderts erlebte einen rasanten Anstieg bei der Erzeugung und Verbreitung digitaler Videos. Dies ist zum Großteil den enormen Fortschritten bei der Aufnahme- und Verarbeitungstechnik geschuldet. War die Aufzeichnung und Verbreitung von hochauflösenden Videos bis vor wenigen Jahren noch einer kleinen Zahl professioneller Produktionen vorbehalten, so kann heutzutage die breite Öffentlichkeit auf die notwendigen technischen Möglichkeiten zurückgreifen. Unabhängige Filmemacher und Hobbyanwender haben die notwendigen Werkzeuge zur Erzeugung von qualitativ hochwertigen Videos in der Hand. Für die Erzeugung technisch eindrucksvoller visueller Effekte sind jedoch weiterhin ein großes Maß an Know-How und Ressourcen notwendig. Oft sind professionell eingerichtete Studios mit Beleuchtungsausrüstung eine grundlegende Voraussetzung hierfür. Kameras, Schauspieler, Kulisse und Requisiten müssen darüber hinaus mit gut erkennbaren Markern ausgestattet werden. Es ist auch durchaus üblich, aufwändige 3D-Modelle von Szene und Schauspielern per Hand zu erstellen.

Ziel dieser Dissertation ist die Bereitstellung von Bildverarbeitungs- und Bilderzeugungsverfahren, die zur Umsetzung visueller Effekte weit weniger manuelle, zeitaufwändige Arbeit am Set und Rechner benötigen. Filmemachern wird es ermöglicht, *Free Viewpoint Videos* einer Szene zu erstellen. Dabei kann die Szene von neuen Perspektiven betrachtet werden, die von den ursprünglichen Kameras nicht abgedeckt sind. Hierzu werden lediglich einige handelsübliche Camcorder benötigt, die die Szene zeitgleich aus verschiedenen Blickwinkeln aufnehmen. Im Anschluss werden die neuen, benutzerdefinierten Perspektiven am Rechner erzeugt. Durch die Verwendung von bildbasierten Syntheseverfahren wird die Notwendigkeit der potentiell

fehleranfälligen Rekonstruktion von Szenengeometrie umgangen. Kalibriermuster, Laserscanner, kontrollierte Beleuchtung oder hardwaresynchronisierte Kameras sind ebenfalls keine zwingende Voraussetzung mehr. Sogar Außenaufnahmen können ohne größeren Mehraufwand realisiert werden.

Im ersten Teil dieser Dissertation wird das automatische Verfahren zur Bildkorrespondenzschätzung vorgestellt. Im zweiten Teil wird das Bildsyntheseverfahren präsentiert. Zur Veraunschauung der Praxistauglichkeit dieser Verfahren stelle ich den Einsatz im preisgekrönten stereoskopischen *Free Viewpoint Video* "Who Cares" der Gruppe Symbiz Sound vor. Auf den Erfahrungen dieses Projekts aufbauend stelle ich einen hybriden Ansatz zur Bildsynthese vor: Dieser vereint die Vorzüge von bild- und tiefenbasierten Verfahren und erschließt neue Anwendungsgebiete wie Blickpunktextrapolation, Bildstabilisierung, Panoramaerzeugung und Relighting.

to Kathrin

Acknowledgements

First of all, I would like to thank my supervisor Prof. Marcus Magnor. No matter if results looked abysmal, papers got rejected, or projects just did not seem to come to a good end, he always supported me. The same is true for my colleagues. Kai Berger allegedly had the hardest job, since he had to share the office with me for several years. I would like to thank Timo Stich and Christian Linz for making me join their work on free viewpoint video and for all their counsel and advice. I am very grateful for the help of Felix Klose and Kai Ruhl, they were an amazing team. I would also like to thank all other researchers who collaborated with me. This includes (but is not limited to) Martin Eisemann, Georgia Albuquerque, Thomas Neumann, Lorenz Rogge, Michael Stengel, Anita Sellent and Benjamin Meyer. Thanks also go to all students who endured my supervision.

I am very thankful that I could participate in the "Who Cares" project and that I could collaborate with Andreas Melcher, Christian Meyerholz, Sebastian Meyerholz and Prof. Uli Plank. I would also like to express my gratitude to Prof. Christian Berger for his support as a friend and colleague. For agreeing to be my the co-referee of this thesis, I would like to thank Prof. Adrian Hilton.

I would like to acknowledge Dan Vulanovic for his 3D wooden manikin "Iggy" that I use throughout Chapter 2.

Last but not least I would like to thank my family. My wife Kathrin and my children Dajana, Benjamin and Jonas had to endure a lot to make this thesis possible. Thanks also go to my parents Lydia and Eduard, and to my brother Adam who have always supported me.

Contents

1

Introduction

Since the invention of cinematography, film enthusiasts have been striving to overcome the technical limitations to create more compelling imagery. Most inventions have focused on reproducing the original scene as faithfully as possible. Early examples are the reproduction of sound, color or stereoscopic views. More recently, digital projections in super-high resolution have been conceived. Recording, playback and even broadcast of 8k imagery is nowadays possible [212]. Just as fascinating as the reproduction of actual actors and objects are the possibilities to create objects that cannot be captured with an actual camera. Since the 1990s, computer-generated images have been used extensively for this purpose: Examples are the dinosaurs in "Jurassic Park" [174] or the flying spaceships in "Independence Day" [44]. Since these early examples of computer-generated content, the available hard- and software systems have progressed considerably. Many visual effects that have traditionally been realized with miniatures or stunt doubles are nowadays created with the aid of computers.

One reason why computer-generated imagery is appealing to film makers is the possibility to re-render the scene from any arbitrary viewpoint. With a computer-generated scene, fast or impossible camera movements, time-freeze shots or extreme slow motion are just as easy to generate as a plain, static camera. Attempts to create these effects with actual sets and actors require painstaking labor in pre-production and on-set preparations. One of the most famous examples are the "bullet-time" effects of the motion picture "The Matrix" [193] where time-freeze and slow-motion effects were combined with an apparently fast-moving camera. Several hundred precisely triggered cameras were employed to create the illusion of this apparently freely moving film

camera. From the present perspective it may seem more feasible to create this kind of effect in a purely virtual environment. However, modeling, animating and rendering a real-world scene is not a trivial task and requires time and money. Furthermore, it is very challenging to maintain the original look of the actual scene.

With the *Virtual Video Camera* project, the possibility to re-render an actual real-world scene from arbitrary viewpoints is provided. Using only a handful of potentially uncalibrated cameras, algorithms and tools for image-based free viewpoint video are developed: The user may create novel camera paths in the post-production stage, i.e., after the material has been captured. As a basis for the renderer, I employ image warping techniques that use bidirectional correspondence maps as input. This allows to jointly interpolate in space and time. Unlike other methods, precisely triggered or calibrated hardware is not required, making ad-hoc captures and outdoor recordings possible.

The scientific contributions of this thesis and the benefits for visual effects production are given in Section 1.1. The overall structure of the thesis is described in Section 1.2.

1.1 Main Contributions

Since the *Virtual Video Camera* is a collaborative project, it is important to highlight the individual contributions of this thesis and to distinguish them from other research conducted by my colleagues. In this thesis, the algorithmic core of the *Virtual Video Camera* project is described. Emphasis is also put on integrating the *Virtual Video Camera* into an actual video production pipeline that also includes the user interface for authoring visual effects. In detail, the main contributions of my work are:

- The long range correspondence estimation described in Chapter 3.
- The image-based free viewpoint rendering formulation along with the constraint tessellation described in Chapter 4.
- The extension to depth-image-based stereoscopic rendering in Section 4.4.
- The workflow integration into an actual video production in Section 4.7.

- The formulation of a hybrid correspondence and depth-based rendering approach in Chapter 5.

- The joint iterative refinement of depth and correspondences in Chapter 5.

Parts of this thesis have been published in peer-reviewed journals and conference proceedings. The long-range correspondence estimation (Chapter 3) was presented as a poster at SIGGRAPH 2010 [236], at the Conference for Visual Media Production (CVMP) 2010 [237], and it was an invited paper in the Journal of Virtual Reality and Broadcasting [238]. The core rendering algorithm (Chapter 4) was presented as a SIGGRAPH 2009 poster [234], published in Computer Graphics Forum (CGF) [235] and an invited presentation at Eurographics 2011. The extension to depth-image-based stereoscopic rendering (Section 4.4) as well as the workflow integration into an actual video production pipeline (Section 4.7) was presented at CVMP 2011 [232] and at FMX 2012 [83]. Some minor aspects of the *Virtual Video Camera* project have also been presented at conferences and workshops, e.g., the simplified acquisition setup [233] or the acceleration of the off-line camera calibration [231], but are beyond the focus of this thesis.

I also contributed to several other research topics related to the *Virtual Video Camera* project that have been addressed by my colleagues, but that are not part of this thesis. I would like to refer the interested reader to the according publications:

- The correction of dense correspondence maps is a crucial element for situations where automatic reconstruction fails. We presented solutions to guide or correct the correspondence search in [224, 247].

- An extension to the original rendering framework was presented in [225, 227]. Instead of modifying and rendering the original images, the renderer operates in the gradient domain. In the same project, SIFT descriptor-based correspondence estimation using a variational framework was proposed.

- The rendering of space-time visual effects, e.g., time blur, space blur, multiple exposures and non-photorealistic effects, was suggested in [228].

- The integration of other depth-based capture devices was proposed in [246, 248].

- First successful attempts to accurate, quasi-dense scene flow reconstruction from non-synchronized cameras were described in [221].
- A real-time media player with free viewpoint capabilities has been implemented and documented in [241, 242].

1.2 Outline

In Chapter 2, some basic free viewpoint fundamentals will be revisited before enumerating and classifying free viewpoint systems. The correspondence estimation algorithm, which is the core element of the image-based renderer, is presented and discussed in Chapter 3. The correspondence-based rendering (CBR) framework can be found in Chapter 4, along with the extension to stereoscopic rendering and the integration into an actual video production pipeline. A hybrid correspondence and depth-image-based renderer (CDIBR) which extends the initial rendering formulation is presented in Chapter 5. Along with the extended capabilities of the renderer, a joint optimization scheme for correspondences and depth is proposed and validated experimentally. A discussion of the contributions as well as concluding remarks draw this thesis to a close in Chapter 6.

2

Related Work

In order to understand the technical details of the different image processing steps, some fundamental prerequisites are covered in this chapter. After giving a brief introduction to the plenoptic function, I survey fundamental image processing techniques, such as feature matching, structure-from-motion, optical flow and depth reconstruction in Section 2.1.

Over the last two decades, several approaches have been introduced to create image-based free viewpoint video. The most fundamental design decision for a free viewpoint system is the selection of an underlying scene model. Depending on the selected model, particular rendering algorithms synthesize an image from a novel viewpoint. It is important to bear in mind that different scene representations are well-suited for different kinds of scenes and capture modalities. Common scene representations and rendering algorithms are presented and discussed in Section 2.2.

With the turn of the millennium, free viewpoint rendering techniques became an established part of visual effects production: In Section 2.3 I highlight feature films, commercials as well as live broadcasts of sports events that make use of free viewpoint rendering.

I identify open challenges in free viewpoint rendering in Section 2.4. How my research on the *Virtual Video Camera* project addresses these challenges, and extends the capabilities of current state-of-the-art approaches, is described as well.

2.1 Prerequisites

Rendering novel views of real-world scenes requires measuring, storing and retrieving the intensity of any light ray going through the scene. All rays of a given perspective view pass through the projection center at world space position $\mathbf{p} = (p_x, p_y, p_z)$. The individual rays can be distinguished by their direction (θ, ϕ), their wavelength λ, and recording time t. The plenoptic function

$$P = P(\theta, \phi, \lambda, t, p_x, p_y, p_z)$$

represents the intensity of any given light ray and can therefore theoretically be used to synthesize any desired novel view of the scene [3]. Unfortunately, this would require dense sampling of the scene which is, in general, infeasible for real-world scenarios. However, there exist a number of strategies to reduce the dimensionality. Considering the tristimulus working principle of the human visual system, it is unnecessary to consider the continuous wavelength domain. Instead, light ray intensities are evaluated independently for red, green and blue light, effectively reducing the dimensionality of the plenoptic function to six. When observing single time instances or static scenes, one can neglect the temporal dimension t. Still, the task of sampling and reproducing of a 5-dimensional function remains. As we will see in Section 2.2, free viewpoint rendering techniques can cope with this challenge by further reformulating the plenoptic function and by synthesizing unknown rays.

Typically, some processing of real-world data is necessary before free viewpoint rendering can be applied. Often, the exact orientation and internal parameters of a camera have to be determined by feature matching (Section 2.1.1) and calibration (Section 2.1.2). In order to synthesize novel viewpoints, certain properties of the original images have to be reconstructed, i.e., pairwise optical flow (Section 2.1.3) or per-pixel depth (Section 2.1.4).

2.1.1 Feature Detection, Matching and Tracking

A feature is a location in an image that has some unique properties. These properties can be used to robustly identify this feature in other images, possibly taken from a different perspective or at a different point in time. This process can usually be broken down into two steps. First, unique features are identified in a feature detection stage.

Second, a descriptor is assigned to every feature point. Feature sets from different images are compared, and a feature matching algorithm is used to pair similar features across images.

The KLT tracker [120, 187, 166] initially searches for features that have a high variance along all spatial dimensions: It only accepts features at image positions where high eigenvalues are present in the covariance matrix. The basic assumption for feature matching is that features are tracked along frames in a video stream and frame-to-frame displacement is small. To track a feature to the next frame, the sum of squared differences of a window surrounding the feature is evaluated. Assuming locally linear gradients, the window is iteratively displaced and re-evaluated until a satisfactory displacement is found.

This basic assumption for feature tracking is often violated, e.g., when frame-to-frame displacement is high or when unordered image collections are used instead of video streams. A common solution is to detect and describe features individually and to find a matching function that robustly pairs similar features. SIFT (Scale Invariant Feature Transform) is one of the most popular techniques for this task [119]. Features are detected by computing Differences of Gaussian convolutions on different scales. Extrema in this scale space are possible candidates for SIFT features. A final selection is made after discarding low contrast and edge-like candidates. Features are identified by the distribution of local gradients. Relative to the predominant local image gradient, 16 local histograms of gradients are stored. In the original implementation, each histogram has 8 bins, resulting in a 128-dimensional feature vector. Feature vector sets from arbitrary image pairs are matched by comparing the best to the second best match using the L_1 distance. Similar to SIFT, several other descriptors exist that can perform matchings between arbitrary image pairs, e.g., MOPs [21], Daisy [186] or SURF [14]. For real-time application, binary descriptors such as BRISK [92] or BRIEF [25] were recently proposed.

2.1.2 Camera Parametrization / Calibration

For most computer vision tasks, the pinhole camera model is used to represent an actual real-world camera. It is defined by its extrinsic and intrinsic parameters. The extrinsic parameters may change from frame to frame for non-stationary cameras and include translation and rotation. Translation is often stored as a three-dimensional vector. To

encode of the rotation, Euler angles, rotation matrices or quaternions can be used. The intrinsic parameters of a camera can be assumed as constant for any given camera (except for zoom lenses): The focal length is either a scalar value or a two-dimensional vector if focal length is estimated independently in x- and y-direction. The principal point is the point on the image plane that is intersected by the optical axis. Often, it is implicitly assumed to be the exact center of the image plane. Also, radial and tangential lens distortion are intrinsic parameters. For the sake of simplicity, lens distortions will be neglected throughout this thesis and the assumption is made that distortion-corrected images are used. The remaining intrinsic and extrinsic parameters are jointly encoded in the projection matrix $\mathbf{C}$. Given a view A, a point in world space X and its projection $\mathbf{x}_A$ (both in homogeneous coordinates) onto the image plane of A, the following relationship holds:

$$d(\mathbf{x}_A) = \mathbf{C}_A \, X \tag{2.1}$$

where d is the scene depth for pixel $\mathbf{x}_A$. When reconstructing parameters for a set of given cameras, a typical approach is to match corresponding features $\mathbf{x}_i$ between images and jointly reconstruct the camera parameters $\mathbf{C}_j$ along with world space positions X_i. Bundler calibration [173] solves this task by selecting a promising initial camera pair A, B that has a large set of matching feature pairs $(\mathbf{x}_{iA}, \mathbf{x}_{iB})$. Using the Levenberg-Marquardt non-linear optimization scheme [137], camera parameters $\mathbf{C}_A$, $\mathbf{C}_B$ as well as world space positions X_i are estimated that minimize the average projection error e_{repr}:

$$e_{repr} = |\mathbf{C}_A(X_i) - \mathbf{x}_{iA}|_2 \tag{2.2}$$

Iteratively, new views are selected heuristically and added to the reconstruction. In order to cope with the increasing amount of optimized values, sparse bundle adjustment [118] is used during the iterative optimization stage.

Instead of iteratively adding images to the reconstruction, subsets of the original data set can be calibrated in parallel and then fused into a single reference model. This can lead to a remarkable increase in computational performance and can further help to maintain robustness, as demonstrated by the Samantha reconstruction pipeline [57].

2.1.3 Optical Flow and Image Correspondences

The typical aim of optical flow computation is to determine the flow of apparent motion between two images I_A, I_B. It is assumed that between the capture of both images a certain amount of time has passed during which scene content and observer have moved arbitrarily, $t_A < t_B$. For any image location $\mathbf{x}$ in A, a vector $\mathbf{w} = (u, v)$ is obtained that describes its displacement towards the corresponding pixel position in image B. Often, (u, v) is referred to as displacement, correspondence or flow vector. Typically, constant brightness is assumed for corresponding image locations, so that:

$$\frac{\delta I}{\delta x}\frac{\delta x}{\delta t} + \frac{\delta I}{\delta y}\frac{\delta y}{\delta t} + \frac{\delta I}{\delta t} = 0 \tag{2.3}$$

where $I_{x,y,t}$ is a function for image brightness at pixel location (x, y) at time t and $\frac{\delta x}{\delta t} = u$, $\frac{\delta y}{\delta t} = v$. Several strategies have been proposed to solve this ill-posed problem. Horn and Schunk [74] introduced an additional constraint. In order to find a globally smooth solution, they suggested to enforce smoothness in the flow field. This is achieved by minimizing both the brightness constancy as well as the Laplacian of the displacement field in an iterative fashion. In contrast, Lukas and Kanade [120] obtain flow vectors for every pixel by finding an optimal solution for each pixel individually. Their method can find highly accurate displacements for well-textured regions. However, their method fails to fill in valid values in untextured or noisy regions since the correspondence vector is optimized only locally.

Different variations of these two approaches have been proposed to solve the optical flow problem. In order to cope with displacements larger than a single pixel, multi-scale search methods have been designed [45, 12]. Although larger displacements can be handled by this extension, small objects may get lost at coarse resolution levels of the image pyramid. Brightness changes can be dealt with by assuming constant gradients instead of constant brightness [22], invariance towards transformations can be achieved by observing SIFT descriptor outputs instead of image brightness [225]. Enforcing symmetry of bidirectional flow maps is also a good strategy to increase the robustness of optical flow computations [6]. A database of state-of-the-art techniques is maintained by Baker et al. [10, 9].

Recently, discrete optical flow estimation strategies have gained attention in the research community [47]. Although their accuracy is limited by the finite label space,

they do not suffer from common drawbacks encountered in long-range optical flow. The main advantage is that high-frequency details typically do not get lost in the image pyramid and that, potentially, a full image search can be performed on the original resolution [115]. The main drawback is the huge demand for memory and processing time and the necessary discretization of label space. In order to cope with the computational complexity, a decoupling of 2D flow estimation a coupled horizontal and vertical search has been proposed [115]. If continuous values are the desired output of the optical flow computation, a continuous extension to these techniques has been proposed [52].

Another research direction are perceptually motivated correspondence estimation techniques. In contrast to defining a correspondence vector per pixel, Stich et al. [177, 178, 179, 176] focus on the visually dominant edges in images. They perform a bipartite matching of edge pixels and propagate the corresponding information to visually similar regions. This techniques exploits the fact that visual artifacts often manifest themselves in visually salient regions.

2.1.4 Depth Reconstruction

Depth reconstruction aims to recover the distance between an observer A and objects depicted in an image I_A taken from the position of view A. Various ways exist to reconstruct depth. Some of them require a controlled lighting environment (e.g., structured light [4, 152] and photometric stereo [200]). Other assume at least knowledge of the illumination (e.g., shape from shading [73]) or introduce additional light sources (e.g., flash photography [58]).

I would like to focus on those techniques that do not make any additional assumptions about the capture environment, but that solely rely on the captured data for reconstruction. The reconstruction of depth using two cameras is referred to as depth from stereo. A database of state-of-the-art reconstruction methods along with quantitative evaluation is maintained by Scharstein et al. [151]. Techniques that accept more than two images as input are referred to as multi-view stereo algorithms [158]. Several output formats are conceivable for depth reconstruction. An obvious depth encoding is to store a single depth value per pixel of each input image. Multiple (depth) images can also be fused into a single layered representation of the scene [165]. If the reconstruction is not dense, point or patch clouds are a valid representation of the scene [50]. Some

algorithms also explicitly reconstruct other geometric properties, e.g., surface orientation and visibility [89, 50] or 3D velocity [191] and store it along with the depth data. Higher level scene models such as surface geometry can also be fitted to the processed depth data [18, 82].

All these basic computer vision tasks are strongly related. Optical flow can be considered a dense feature tracking task. In fact, the KLT tracker has been used for both sparse feature tracking and dense optical flow estimation. Depth reconstruction can be considered as a special case of optical flow: In case of rectified images, depth is equivalent to inverse disparity, i.e., horizontal displacement of corresponding image features. Depth estimation is therefore equivalent to optical flow estimation that is constrained to horizontal flow vectors. Similarly, optical flow between two images can be obtained by reprojection if scene content did not move during the capturing process and both depth values and camera parameters are known.

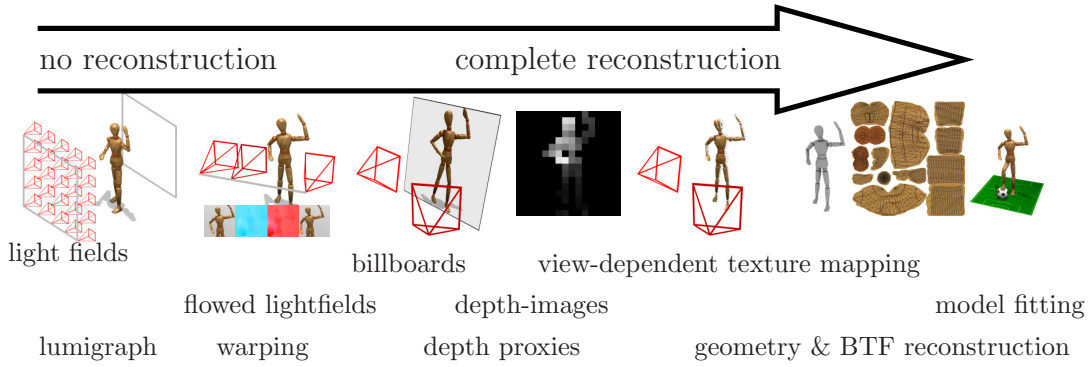


Figure 2.1: Overview of image-based free viewpoint systems. While some approaches rely on a dense sampling of the scene (far left), others rely on high-quality reconstruction and geometric scene models (far right). Most free viewpoint video techniques can be located somewhere between these two extreme scenarios.

2.2 Free Viewpoint Video Systems

The common aim of free viewpoint video is to re-render real-world scenes from novel viewpoints. There exist purely image-based approaches that merely re-sample the captured data. The other extreme are geometry-based techniques that reconstruct a high-level model of the scene. As proposed by Lengyel [91], most free viewpoint methods can be positioned in a continuum between these two extrema, Fig. 2.1. I provide an

up-to-date and thorough categorization of different basic techniques and real-world applications. Other surveys of free viewpoint rendering have been conducted by Shum and Kang [167], Smolic [170], Linz [98], and Germann [54].

2.2.1 Light Fields and Lumigraphs

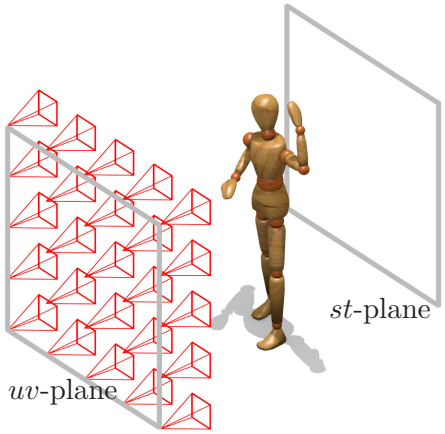


Figure 2.2: Light field capture. A dense grid of cameras is placed on the uv-plane. They capture light rays passing through the space between the uv- and st-plane. If the scene content remains static, captures can be achieved with a motorized camera gantry.

The basic idea of light field rendering [94] is to reformulate the plenoptic function as a four-dimensional lookup task, Fig. 2.3. By assuming an unobstructed view between the observer and a camera plane (also referred to as uv-plane), each light ray passing through the uv-plane can be parametrized by four scalar values (u, v, s, t). Values (u, v) are determined by intersecting a viewing ray with the uv-plane. Parameter values (s, t) are similarly obtained by intersecting the viewing ray with a (virtual) focal plane (also called st-plane). Construction of the light field is done by placing a dense 2D array of cameras on the uv-plane. The four corners of the camera frustum must be aligned with the corners of the st-plane, Fig. 2.2. To prevent discretization artifacts, filtering schemes can be employed.

Several modifications and extensions to the original algorithm have been proposed. In Lumigraph rendering, capturing the scene is simplified by allowing a more unconstrained placement of cameras [65]. Before rendering, all image data is transferred to

the (u, v, s, t) domain in a rebinning step. Davis et al. [37] follow a similar idea that allows light field capture with commodity hardware, i.e., mobile phones.

Advantages. Light fields do not rely on any form of scene geometry or optical flow reconstruction. As long as camera calibration is precise and the sampling rate is high enough, high quality rendering is possible. Since the plenoptic function can be effectively reduced to four dimensions and adequate compression schemes exist [122], storage and streaming of light fields is feasible [188].

Disadvantages. The amount of data necessary is still immense and usually limits the viewpoint range drastically. The capture hardware has to be precisely calibrated and the capture process may take a long time if the number of desired input images is higher than the number of available cameras. In the latter case, the scene must also remain static during acquisition. Special hardware has been suggested to capture light fields with a single chip [145] and recently, similar devices have been developed into commercial products [121, 198]. With single chip systems, there is a trade-off between image resolution (uv-resolution) and possible viewpoint variations (st-resolution).

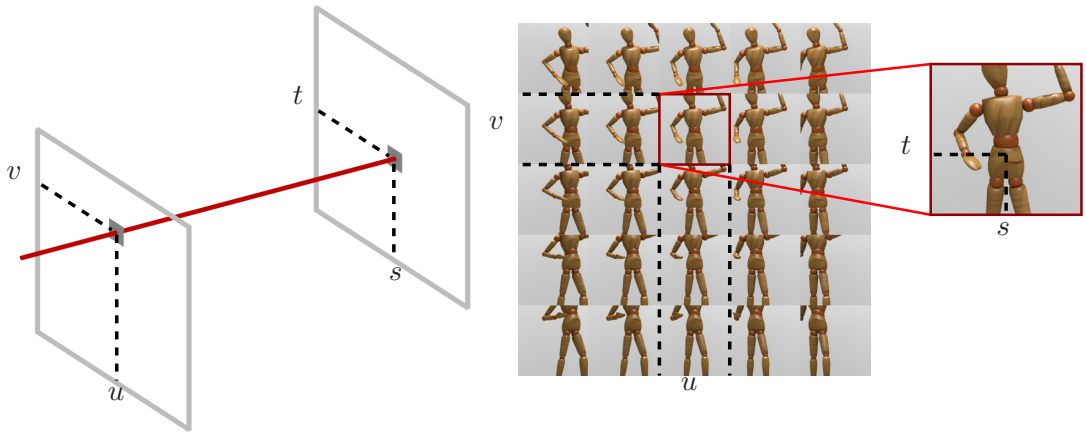


Figure 2.3: Light field rendering. For each rendered light ray, an intersection test with the uv- and the st-plane is performed (left). The obtained coordinates are used for a lookup of the 4D light field data (right). Virtual views are obtained by ray tracing intersection rays for every pixel location.

2.2.2 Flowed Light Fields

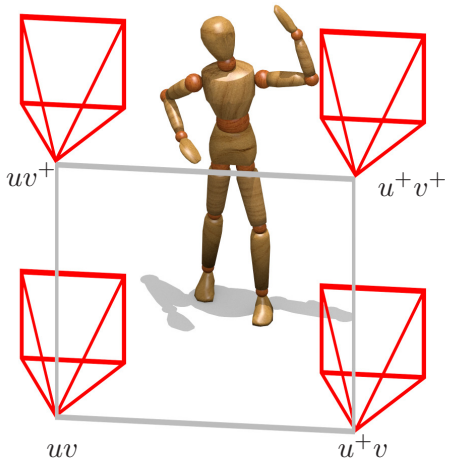


Figure 2.4: Flowed light field capture. The set-up is similar to the one used by light fields. The major difference is that the distance between cameras is larger (less cameras are required to cover the same uv-plane). In this toy example, only four cameras are used. This is compensated by the warping scheme in the rendering stage.

Flowed light fields use a rendering scheme that is very similar to light fields [42]. In order to cope with a much sparser sampling of cameras, image warping is used to synthesize cameras on the uv-plane, Fig. 2.4. Analogous to light field rendering, each ray is intersected with the uv-plane. Instead of simply looking up the nearest camera on the plane however, all four cameras are evaluated that span a rectangle around the intersection point, Fig. 2.5. The bilinear coefficients of the point within the rectangle are evaluated. Along with precomputed optical flow fields between the original images, a backward warping scheme [10] is applied to obtain the corresponding pixel positions in the four original images. The final color for the ray is computed by weighting the four values obtained by the lookup.

The capturing process is simplified in two ways: First, the sampling is much less dense. For a full 360° surround capture, 3×30 captured images can be sufficient for a scene containing a single actor [42]. Second, the warping scheme enables the renderer to cope with slightly misaligned images. Therefore, it is not necessary to capture all images at the same time instant. Einarsson et al. [42] exploited this property of flowed light fields to capture cyclic motions of human actors with only three cameras, each time from a different viewing angle.

Advantages. Flowed light fields require much less input images than standard light fields. In addition, the warping scheme can account for small errors, e.g., calibration errors or unsynchronized recording equipment.

Disadvantages. Flowed light field rendering relies on computing dense flow fields. If the distance between cameras is large or the scene composition is complex, automatic optical flow estimation is a hard and error-prone task. Additionally, the backward warping scheme cannot cope with object occlusions. This causes ghosting artifacts in the final rendering.

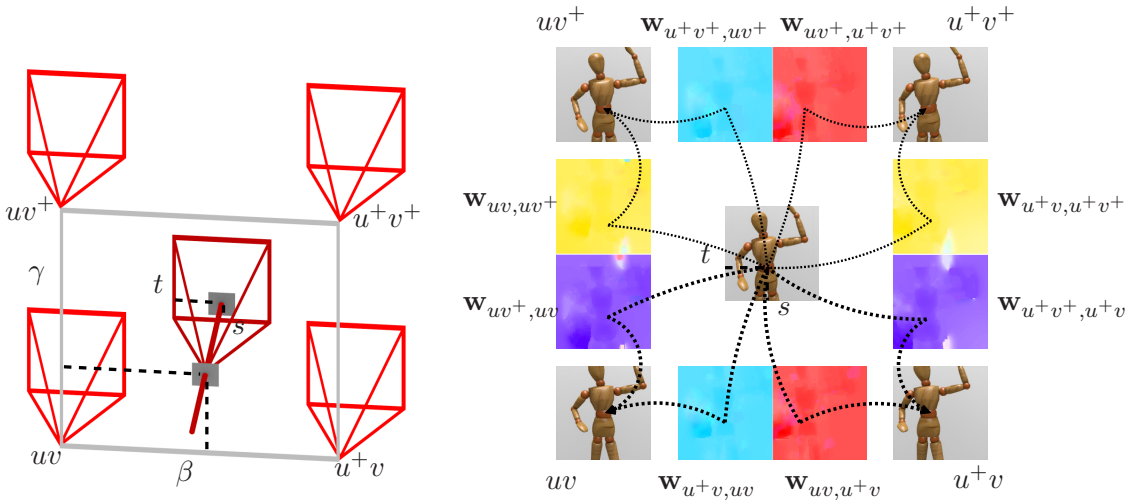


Figure 2.5: Flowed light field rendering. For each viewing ray that intersects the uv-plane, the four surrounding images $u[^+]v[^+]$ along with bilinear weights β, γ are obtained. Prior to rendering, optical flow fields $\mathbf{w}_{u[^+]v[^+],u[^+]v[^+]}$ are computed between horizontal and vertical neighbors. During rendering, a backward warp $\mathbf{w}$ is applied to obtain the corresponding pixel color at each of the four original images. The resulting color is a weighted blend of the four original colors of the input images.

2.2.3 Warping-Based Approaches

In warping-based approaches (also known as image morphing or correspondence-based rendering), synthetic in-between views are generated using the original images A, B and bidirectional flow fields $\mathbf{w}_{AB}, \mathbf{w}_{BA}$, Fig. 2.6. This method is not necessarily restricted to view interpolation but has a long-standing tradition for creating transitions between similar images: For example, it has been used to create transitions between different actors, who are performing an identical choreography [16]. An extension to more than

two images has been proposed by Lee et al. [88]. Chen and Williams [33] propose to use image warping for viewpoint interpolation. In order to re-render scenes from in-between viewpoints, they warp the original pixel locations according to the flow vectors. Image warping is repeated for each source image. The final result is a weighted blend of all warped images. The combination of image warping and blending is also referred to as image morphing. Relying on synthetic scenes, the flow vector for a pixel location $\mathbf{x}$ can easily be derived if per-pixel depth and camera matrices are known:

$$\mathbf{w}_{AB}(\mathbf{x}) = \mathbf{C}_B(\mathbf{C}_A{}^{-1}(\mathbf{x})) \tag{2.4}$$

where $\mathbf{C}_A{}^{-1}$ is the inverted projection matrix of camera A. In order to cope with arbitrary camera orientations, Seitz and Dyer [159] propose a pre- and post-warping scheme that guarantees to produce geometrically valid in-between views. Saito et al. [149] automatically reconstruct the 3D geometry of a scene captured with 49 cameras. Following Chen and Williams [33], they obtain depth and correspondence maps from the 3D scene and use image morphing for view synthesis. McMillan and Bishop [127] apply a similar warping scheme to real-world panoramic views. Levieux et al. use a warping-based rendering for producing free viewpoint video of objects with repetitive motions [93]. A view synthesis for uncalibrated footage was presented by Fusiello [51].

Similarities exist to flowed light fields (Section 2.2.2) as well as to depth-based rendering (Section 2.2.5). If a backward warping scheme is used, i.e., if each pixel of the synthetic view is queried for its location in the original image, image warping is a special case of the flowed light fields: The center of the virtual camera is located between the original camera positions, or, if more than two cameras are used for image synthesis, on a manifold spanned by all camera locations.

If forward warping is used, i.e., if each pixel of an original image is queried for its (forward warped) location in the synthetic image, warping is akin to depth-based rendering: If both input images A, B are rectified and are captured at the same point in time, all flow vectors have only a horizontal component. This horizontal flow, which is often referred to as *disparity*, is proportional to the inverse depth at a given pixel position.

Advantages. Analogous to flowed light fields, warping-based rendering does not rely on as many input images as light fields do. The amount of image data is potentially

even smaller since cameras do not have to span a 2D manifold, but can be arranged in arc-like set-ups around the scene. They can also cope with inaccurate calibrations and unsynchronized input as long as valid correspondences can be obtained.

Disadvantages. The quality of the rendered output depends on the availability of plausible correspondence maps. The placement of a virtual camera is constrained to the manifold spanned by the cameras. Occlusions are also a problem for warping-based approaches. Although purely image-based occlusion detection is an active research area [80, 71], some algorithm use additional depth information to resolve occlusion ambiguities [33].

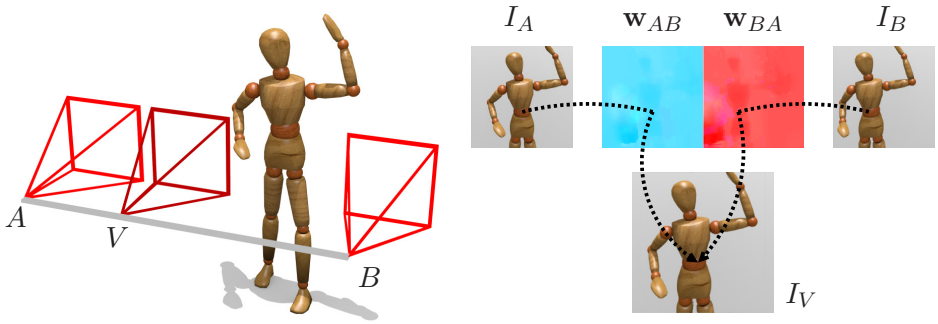


Figure 2.6: Warping-based free viewpoint video. At least two images I_A, I_B contribute to the synthetic rendering I_V. For each pixel in I_A and I_B, the image fragment is displaced according to the flow vectors in $\mathbf{w}_{AB}$ and $\mathbf{w}_{BA}$, respectively. This displacement is often referred to as forward warping. The final image is a blend of the two aligned source images. An alternative method is to apply a backward warping scheme where the flow vector lookup is performed in the synthesized view I_V.

2.2.4 Geometry Proxies

Geometry proxies of a scene's surface geometry allow for visually convincing rendering even if scene geometry is hard or impossible to reconstruct. The most basic approximation is a single fronto-parallel plane (sometimes referred to as *billboard*) in the middle of the scene, Fig. 2.7. A novel viewpoint is rendered by projecting the original image onto this plane and rendering the textured plane from a novel viewpoint. Cross-blending

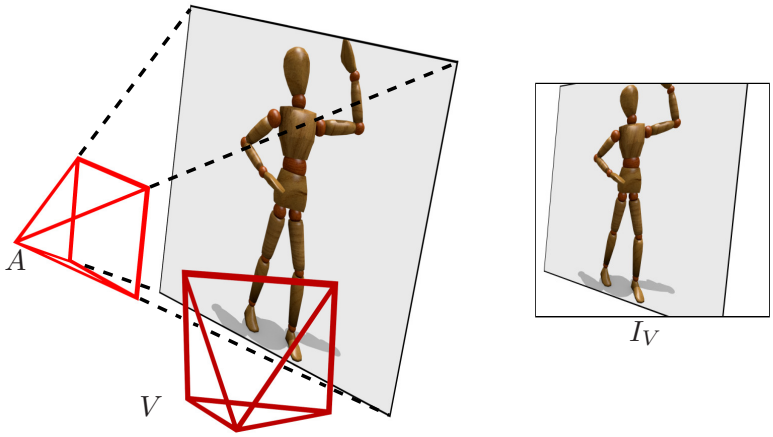


Figure 2.7: Geometry proxy-based rendering. Only a coarse geometric representation of the scene is required. In the depicted case, it is represented by a single view-dependent, fronto-parallel billboard. During rendering, the source image is projected onto the proxy geometry.

between two views can be used to interpolate between them [173, 172]. Images can also be divided into several planar regions to achieve a more accurate representation. This approach has been successfully applied to football players [55] and architectural scenes [156]. Depending on scene content and capturing modalities, it can be feasible to use a more detailed proxy representation. Different approaches use thousands [203, 61] or hundreds of thousands [75] of microfacets or particles to represent the scene. Further rendering improvements can be achieved by faithfully reconstructing the boundary colors of neighboring proxies [55], by merging them in image space [75] or by applying local displacements to billboards [196]. If no valid depth information can be obtained for certain image regions, ambient point clouds can be used to visualize this uncertainty [59]: Random depth values are assigned to pixels of unknown depth so that they form an amorphous, unobtrusive point cloud.

Advantages. Depth proxy-based approaches are able to yield visually pleasing results without a complete (i.e., pixel-accurate) depth reconstruction of the scene. Depending on the granularity of the reconstruction, accuracy is traded for robustness: Very coarse representations of the scene (e.g., billboards) can be robustly estimated. Memory requirements are also small due to the limited amount of geometric information. Finer representations (e.g., microfacets) reproduce the geometric details, but are harder to

reconstruct. In contrast to purely image-based approaches (Sections 2.2.1-2.2.3), only a single source image is needed for rendering. The placement of virtual cameras is arbitrary and not restricted to any kind of camera manifold.

Disadvantages. Depending on granularity of the scene representation, rendering artifacts are visible. Especially when using a simple billboard proxy, strong ghosting artifacts are visible when cross-blending images from different perspectives. The impact of perceived image quality has been studied by Vangorp et al. [190]. Although depth information is typically only required on a coarse level, camera calibration and some sort of depth reconstruction are always necessary.

2.2.5 Depth-Image-Based Rendering (DIBR)

In contrast to depth proxies, depth-image-based rendering relies on obtaining depth values for every pixel in all input images. To obtain valid depth values, dense per-pixel stereo matching must be performed, Section 2.1.4. According to the depth value in a reference view A, each pixel is reprojected to a virtual view V [46], Fig. 2.8. Different methods can be employed to re-render the original view. One possibility is to treat all pixels independently and to apply point cloud rendering or point splatting. Some very early results have been presented in the *Immersion* project [206, 2]. Another possibility is to treat the whole source image as a connected mesh [211, 208]. In the presence of large depth discontinuities, single quads of the mesh are locally discarded. Alpha matting is used to estimate local foreground color and alpha values, and an additional boundary layer is rendered that guarantees the smooth transitions between different depth layers.

A very challenging remaining problem is disocclusion handling. Both point splatting and mesh rendering methods may produce holes in the final image [180]. One solution is to use two [211] or more [208] source images for rendering that hopefully fill in the holes in the final image. Another possibility is to use a layered representation of the scene [165, 134]. Still, holes may remain in the synthetic view. Several inpainting techniques have been proposed to assign plausible color information to unfilled regions [35, 133]. Schmeing et al. present a benchmark and survey for common inpainting techniques [153].

Advantages. Since dense depth information is used, the reprojection into the virtual view is possibly highly accurate. Similar to depth proxies, the placement of a virtual video camera is more or less arbitrary. Also, only a single source image suffices for rendering. This holds especially true for layered representations of the scene.

Disadvantages. Accurate, dense depth reconstruction is difficult and error-prone. It requires accurate camera calibration, the scene content must remain static between captures, and the depicted objects must in general have Lambertian reflection properties. Although the placement of virtual cameras is arbitrary, it is practically restricted to positions close to the original viewpoints, otherwise disocclusion handling becomes an infeasible endeavor.

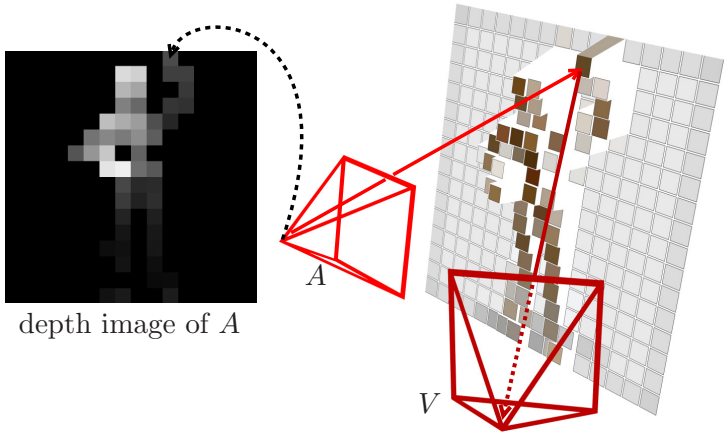


Figure 2.8: Depth-image-based rendering (DIBR). The depth for each pixel is assumed to be known (visualized by black and white depth map, left). According to pixel depth and camera matrix of A, each pixel is projected to its world space position. Using the projection matrix of the virtual camera V, the image is projected from view A to the image plane of V.

2.2.6 Geometry Reconstruction and View-Dependent Texture Mapping

So far, only view-dependent geometric representations of the scene have been considered (Sections 2.2.4-2.2.5). Alternatively, a consistent geometric model can be reconstructed prior to rendering. In the rendering stage, a view-dependent selection of input images is used for projective texturing, Fig. 2.9. If the scene contains a single object

of interest, visual hulls are a common tool to obtain a conservative approximation of the geometry [141, 13, 126]. Of course, other, more elaborate reconstruction schemes exist that do not make specific assumptions about the scene. As proposed in [171] and [104], one can use structure-from-motion calibration [173] before applying quasi-dense multi-view reconstruction of surface patches [50]. A final watertight model can be obtained with Poisson surface reconstruction techniques [82]. Even for scenes that contain non-Lambertian objects solutions exist to obtain a surface representation [192]. For architectural scenes, tools have been proposed that allow for user assistance, e.g., the facade system [39] or more recent, commercial products like sketchup [64].

Advantages. For rendering, only a very limited amount of views is needed to reproduce the original object appearance. If a valid surface mesh can be obtained, the scene can be re-rendered from any viewpoint and virtual cameras are not restricted to any particular area. In addition, the surface model can have additional practical use, e.g., it can receive and cast shadows in a virtual scene.

Disadvantages. As reported by Eisemann et al. [43], visibility and alignment of projected textures are non-trivial problems and disturbing artifacts occur if errors are present in camera calibration or geometry reconstruction. Obtaining a valid surface mesh can be very difficult and error-prone. Many automated solutions rely on silhouette extraction, effectively limiting the applicability to scenes with a single, easily segmentable object of interest. User-assisted methods are by default labor intensive, and most such user interfaces are geared towards architectural scenes [39, 64].

2.2.7 Complete Reconstruction

Free viewpoint video can be achieved by a complete reconstruction of the real-world scene, Fig. 2.10. Not only the surface geometry is obtained (as in Section 2.2.6), but also a consistent, view-independent description of the surface appearance. Surface appearance can be encoded as a diffuse texture map [189, 175] or as a more complex representation, such as bidirectional texture functions [36, 157]. Given a surface mesh, source images and camera parameters, several algorithms exist to create a consistent diffuse texture atlas [195, 90, 53]. Complete, automated end-to-end systems for capture

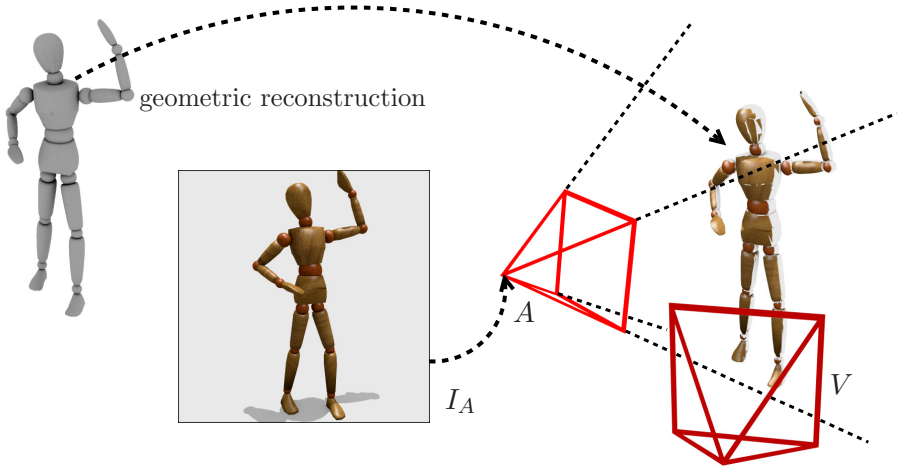


Figure 2.9: Geometric reconstruction and view-dependent texture mapping. A 3D surface model is reconstructed automatically or with the user assistance. From a given reference view, the original image is projected onto the geometry. Since not all visible parts of the object are covered by texture, multiple reference views are typically used for texture mapping. The selection and weighting is assigned based on the orientation of the virtual camera.

and reconstruction of models have been proposed. Starck and Hilton reconstruct a consistent mesh along with a texture atlas from only 8 video streams [175]. They combine visual hull reconstruction and stereo cues into a multi-view reconstruction framework. Their scene representation and compression allows for real-time streaming and rendering with arbitrary bit rates. A similar end-to-end approach that uses quasi-dense point cloud estimation along with Poisson surface reconstruction has been proposed by Liu et al. [117]. Their texture atlas generation relies on obtaining a high quality mesh in the preprocessing stage. Matsuyama et al. developed an end-to-end system that puts emphasis on real-time processing [125]. Schwartz et al. use a capturing dome for small objects with controllable lighting to capture both object geometry and appearance in high quality [157]. Recently, multiple commercial software products have been released that reconstruct a surface mesh along with a texture atlas from several dozen input images [7, 5, 76]. A qualitative evaluation and comparison of these tools has been conducted by Nguyen et al. [136]. If an automatic reconstruction is infeasible, the scene geometry can be reconstructed with the aid of the user. E.g., the videotrace software [189] allows the user to specify points and edges of the mesh to determine 3D-positions semi-automatically.

Advantages. A complete 3D-reconstruction of the scene makes very compact scene representations possible. The source images are not necessary for rendering and can be discarded after the reconstruction phase. Only a single texture atlas has to be created and can be re-used for multiple frames of a video sequence [175]. The scene representation is in general very easy to render, since (graphics) hardware, software and file formats are readily available. Since efficient compression techniques exist, web-based streaming of object and texture data is possible [155].

Disadvantages. Reconstruction of geometry and appearance is a very hard problem and typically requires a controlled capture environment for automated processing [155]. Otherwise, user-assistance is required to obtain a high-quality representation [189]. Although automated schemes have been proposed for general, real-world scenes, they require a high amount of input data and may still not produce artifact-free results [136]. Many approaches restrain themselves to the reconstruction of the Lambertian surface properties (i.e., diffuse reflection). While view-dependent texture mapping (Section 2.2.6) may preserve specular highlights to some extend, these appearance properties are not preserved if a diffuse appearance model is reconstructed.

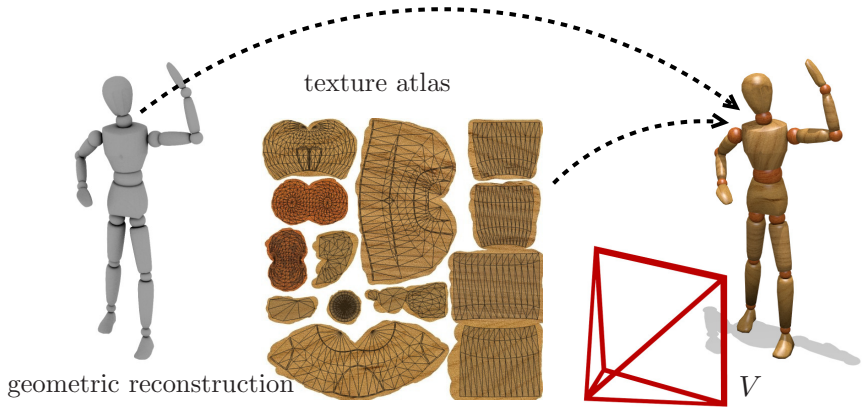


Figure 2.10: Complete Reconstruction. Both surface geometry and appearance (e.g., diffuse texture atlas or BTF) are obtained. In the rendering stage, a compact model representation can be used and the original source images do not have be to accessed.

2.2.8 Scene-Specific Models

In the preceding survey of common free viewpoint techniques (Sections 2.2.1-2.2.7), only techniques have been mentioned that estimate scene models directly from the captured images. Some approaches considerably improve scene reconstruction by employing an a priori model of the scene or its contents. Instead of allowing arbitrary reconstruction, they can enforce consistency with a given model or fit the model to the recorded data. Prominent applications are pose estimation and motion capturing where a skeletal representation of an actor is fitted to match the pose or the movements of its captured counterpart [131]. Possible output data are joint angles, shape parameters and time varying textures for a given hand-modeled 3D actor [27]. Hasler et al. [69] fit a statistical human model [70] to unsynchronized outdoor camcorder recordings. Jain et al. [81] use the same model to estimate poses that were recorded with a single camera. In order to obtain the geometry of individual actors, laser scans [38] or other depth sensors [205, 87] can be employed.

Of course, known geometry and appearance of other scene parts can also be exploited. Football fields, for example, have a well-defined surface (flat) and appearance (green with white markings). This knowledge can be used for background segmentation and rendering [72, 55].

2.2.9 Temporal Interpolation

Many free viewpoint approaches do not explicitly provide any special mechanisms to cope with time-varying data. Although it is always possible to apply any free viewpoint technique independently on consecutive frames of a sequence, it is often problematic. First, temporal coherence is not guaranteed during scene reconstruction, and the independent processing of frames can result in flickering artifacts. Second, temporal interpolation is not possible out-of-the-box.

Furukawa et al. tackle the problem of temporal coherence by obtaining an initial surface representation [50] and tracking it in successive frames [49]. Occlusion of scene parts can lead to holes in the surface model. A temporally coherent mesh completion has been proposed by Li et al. [95]. Instead of enforcing temporal consistency in a post-processing step, Goldlücke and Magnor achieve a temporally consistent surface reconstruction by finding a photo-consistent, spatio-temporal hypersurface [62]. They

Figure 2.11: Scene specific models. Free viewpoint rendering can exploit scene specific a priori information to simplify the reconstruction process. The knowledge of scene geometry and appearance, e.g., recordings of a football stadium, can be used to simplify background segmentation. Other typical a priori models are hand-modeled, laser-scanned or statistical representations of human actors.

present a distributed computation scheme to jointly estimate the surface geometry in a 20-frame multi-view video [63]. Tracking of a consistent facial mesh can be performed with one or several reconstruction anchor frames, i.e., frames with visually similar, neutral facial expressions [19, 15]. The explicit reconstruction of scene velocity, also referred to as scene flow, has been investigated by Vedula et al. [191].

Chen and Williams mention that all light field properties can potentially be interpolated [33], Zhang et al. built on that idea and designed a ray space warping scheme for light field rendering [207]. Many free viewpoint rendering approaches assume that all captured images are precisely synchronized. For many practical applications, this assumption is violated. Often, multi-view camera set-ups are constructed on a low budget. The commodity hardware in such set-ups often provides no means for inter-camera synchronization [69]. Even if high-quality equipment is used, misaligned frames [78] and frame drops [77] may occur. Only few practical free viewpoint algorithms explicitly cope with non-synchronized multi-view captures. Wang et al. use a two-stage approach [194]: First, they perform a temporal interpolation of unsynchronized multi-view recordings. Afterwards, they apply light field rendering for view interpolation. Li et al. deliberately use temporal offsets in their capture to achieve a temporally upsampled interpolation

of their data [96]. But while capture times differ, subsets of cameras still have to be synchronized for the initial spatial reconstruction.

2.2.10 Mixed Approaches and Extensions

The classification introduced in the previous sections aims at giving an overview of fundamental free viewpoint approaches. When looking at some actual free viewpoint systems it is apparent that many of them do not fit precisely into a single category. Some approaches have been proposed that combine different techniques: Unstructured lumigraphs generalize both lumigraphs and view-dependent texture maps [24]. Depending on the level of detail of the proxy geometry, they behave like one of the both extremes or a mixture of them. The view-dependent texture mapping approach of Debevec et al. [39] also employs depth-based rendering at a fine level: For each reconstructed face, the original textures can be projected onto the geometry, and local depth maps can be computed that compensate the remaining projection errors. Yang et al. [204] propose view-dependent textured splatting, a mixture between view-dependent texture mapping and point splatting.

For many practical applications it proves to be beneficial to segment the scene into different regions (e.g., actors and background), and to treat them differently. In outdoor sports scenarios, players are separated from the field at an early stage of the processing pipeline [72, 55]. While billboard representations or 3D surfaces are reconstructed for the individual players, it is often sufficient to represent the playing field by a single plane. For other, more general scenes, a similar scheme can be used: Ballan et al. [11] use a user-supervised segmentation of a foreground actor for various unsynchronized in- and outdoor captures. While the background can be reconstructed in high detail and rendered with view-dependent texture mapping, the foreground actors are robustly handled with a billboard approximation.

2.2.11 Corrections

As each individual rendering method has its own advantages, it also has its inherent drawbacks and failure cases. In some scenarios, the user can assist the reconstruction process until pleasing results are obtained. Several approaches have been suggested that require user input for geometry reconstruction [39, 189]. Others require sparse user input for scene segmentation [11, 68]. Chaurasia et al. [32] incorporate both silhouette

and depth cues from the user for their view interpolation method. Although their warping scheme does not necessarily require user input, the additional cues significantly improve the results. Floating textures [43] provide a correction mechanism that does not require any manual intervention. Prior to the blending stage in image-based rendering, the projections from different source images are aligned using real-time optical flow. Similarily, Germann et al. [56] use sparse feature matches to align the 3D geometry of the scene.

2.3 Commercial Applications of Free Viewpoint Video

It is hard to pinpoint an exact date when free viewpoint rendering became a commercial success or reached a wider audience. Although experimentations with camera arrays can be tracked to the early 1990s [185] and have probably started even earlier, the first popular example of free viewpoint video is the bullet-time effect in the motion picture "The Matrix" of 1999. It was created with a dense camera array and optical flow-based, narrow-baseline image interpolation [160]. Since then, free viewpoint video has had a considerable impact on the movie industry (Section 2.3.1) and also found its way into live broadcasting (Section 2.3.2). On the other hand, a breakthrough of free viewpoint video rendering in computer games and other interactive media is yet to come. Although there have been attempts to use lidar scans and photographs to recreate scenes and actors for games, e.g., race tracks [143], rendering quality does not match traditional content generation. First working prototypes of convincing interactive free viewpoint avatars are currently emerging in the research community [28].

2.3.1 Feature Film and Commercials

Since "The Matrix" movie came out in 1999, camera arrays have been used extensively in creating free viewpoint sequences in feature films and commercials [79, 185]. It is possible to create this kind of effect with physical production tricks. For example, mounts, rigs and wires were used to create the illusion of frozen time in the the TV series "Heroes" [161]. Nonetheless, elaborate techniques have evolved that use lidar-fitted capture stages to record and re-render single actors, as seen in the movie "Hugo" [140]. There have also been attempts to integrate geometry reconstruction-based free viewpoint video into standard post-production pipelines. In the "Midas

Touch" experimental video [182], props and actors were recorded with standard HD cameras, reconstructed with post-processing software and re-rendered with different object materials (e.g., gold). The light stage project by the University of Southern California has enabled the application of free viewpoint video and related techniques to movie production [164]. Both scene lighting and high speed capture hardware can be precisely controlled and allow for a variety of applications. A recent example is the capturing of an actor's facial performance and the performance transfer onto a 3D bust or maquette, as seen in "The Curious Case of Benjamin Button" [162]. Since the maquette is also scanned with the light stage and surface appearance can be reconstructed, the model can be relit to match arbitrary real-world scene backgrounds. A similar approach are virtual productions, where the live action is recorded, the pose of the actors is estimated, and a virtual surrogate is rendered on top of the original frame. A recent example are humanoid robots which were inserted into the live on-set preview during shooting "Real Steel" [163]. The latency between capture and image generation is very short and makes an on-set preview possible [135]. For static scenes, 3D surface reconstruction is nowadays possible in production quality with off-the-shelf hard- and software [139].

2.3.2 Sports and Live Broadcast

Application of free viewpoint video in sports events dates back to the famous EyeVision multi-camera system (a spin-off from Carnegie Mellon University) that was used during the live broadcast of the 2001 Super Bowl XXXV [67]. Over 30 cameras were jointly calibrated and synchronized. While a human operator controls a single reference camera, zoom, focus and rotation are propagated to the other cameras so that they all converge to a single point of interest. In critical situations of the game, e.g., touchdowns or fumbles, the broadcast operator can easily display the football fields from various angles. Although this technology was a break-through for free viewpoint television, it took seven more years until virtual in-between views were shown during a live broadcast. The LiberoVision system, a spin-off from ETH Zürich, renders actual transitions between sparsely placed cameras using billboard representations of players, which was demonstrated during the European football championships 2008 [97]. A similar system has been developed by the BBC, Snell & Wilcox and the University of Surrey [66]. Since then, constant progress has been made in commercial free viewpoint

systems, the most recent example being free viewpoint renderings during the 2012 London Olympics. During the gymnastics competition, an actual 3D representation of the competitors was created on-the-fly and used for view interpolation [144]. For outdoor water sports, a camera array of water-proof GoPro cameras was used for the Ripcurl advertisment campaign [184]. Since the cameras are densely spaced and optical flow is used to compute in-between images, bullet time-like effects can be created of surfers in the open water.

2.4 Open Challenges

As presented in the preceding sections, free viewpoint systems have come a long way and are nowadays common in movie production and broadcast. However, most approaches still require elaborate hardware set-ups to work successfully: Cameras need to be synchronized and/or color-calibrated. Also, background appearance and illumination must be controllable. It would be much preferable to apply free viewpoint video to any kind of scene, regardless of capture modalities and scene content.

There already exist some approaches that try to deal with this so-called "casually" captured multi-view material: The approaches by Ballan et al. [11] and Goesele et al. [59] trade rendering quality for robustness. In image regions where they fail to recover the scene geometry, they use a simplified, potentially inaccurate, representation for view interpolation. Ballan et al. [11] approximate moving actors by single, fronto-parallel billboards, while Goesele et al. [59] assign random depth values to parts of images where reliable depth values cannot be obtained. I strive to find solutions that produce visually convincing free viewpoint video of these kinds of scenes. The *Virtual Video Camera* is designed to cope with non-synchronized capture devices and dynamic scene content. Chaurasia et al. [32] use an elaborate depth reconstruction pipeline for their free viewpoint renderer. To provide high quality view interpolation, their approach requires user input at multiple stages of the reconstruction. The *Virtual Video Camera* aims to rely on only a very limited amount of user interaction. While, in some cases, user input may be acceptable to reach the desired output quality, most scenes should be handled automatically. It is my goal to incorporate all available information to render artifact-free representations of a given scene. Germann et al. [56] use sparse feature matches to align the 3D geometry of a given scene. I want to go beyond this idea and

devise a joint, dense depth- and correspondence-based reconstruction and rendering pipeline. In contrast to their approach, I iteratively refine correspondences and depth and provide pixel-exact information. Furthermore, the *Virtual Video Camera* should be able to interpolate viewpoints in both space and time. This is, in general, not the case in other free viewpoint approaches.

In order to achieve these goals, some challenges considering image processing and rendering have to be faced. Since I do want to accept "casually" captured input, some sort of free viewpoint approach has to be used that aligns the unsynchronized material in the final render. As discussed in Section 2.2.3, correspondence-based renderers fulfill this requirement. Numerous approaches for correspondence estimation exist. In Chapter 3, several state-of-the-art techniques are adapted and combined to provide for a long-range, occlusion-aware, high-resolution **correspondence estimation** algorithm. For **joint spatio-temporal view interpolation**, an appropriate rendering scheme has to be devised. Unlike existing approaches, it should allow to freely viewpoint-navigate in multiple spatio-temporal dimensions. I present this core rendering algorithm along with extensions to stereoscopic rendering in Chapter 4. For increased robustness in image processing, less manual user interaction and improved rendering capabilities, a **hybrid correspondence- and depth-image-based renderer** is presented in Chapter 5. It combines the strengths of existing depth- and correspondence-based approaches. Some very demanding multi-view data sets are, for the first time, processed automatically for free viewpoint rendering.

3

High Resolution Correspondences for Image Interpolation

Establishing dense image correspondences between images is still a challenging problem, especially when the input images feature long-range motion and large occluded areas. With the increasing availability of high-resolution content, the requirements for correspondence estimation between images are further increased. High resolution images often exhibit many ambiguous details, where their low resolution predecessors only show uniformly colored areas, thus the need for more specific and robust matching techniques arises.

I present an approach for establishing dense pixel correspondences between two images of up to 1920×1080 px resolution. I pick up on the idea of Liu et al. [115] to incorporate dense SIFT feature descriptors [119] for pairwise image correspondence estimation. While Liu et al. identify visually similar regions in low-resolution images, SIFT features are used as a descriptor for fine detail in high-resolution images. The approach provides a versatile tool for various tasks in video post-production. Examples are image morphing, optical flow estimation, stereo rectification, disparity/depth reconstruction and stereo baseline adjustment.

The contribution to the research area of correspondence estimation is the integration of various features and improvements into the optimization framework: In order to match fine structural detail in two images, I compute a SIFT descriptor for each

pixel in the original high-resolution images. To avoid ambiguous descriptors and to speed up computation, each image is downsampled by selecting the most representative SIFT descriptor for each $n \times n$ grid cell (typically $n = 4$). An initial lower resolution correspondence map is then computed on the resulting downsampled versions of both images. Since the original Belief Propagation implementation by Felzenszwalb et al. [47] might not retain crisp borders due to the grid-based message passing scheme, a non-grid-like regularization technique as proposed by Smith et al. [169] is employed. As memory consumption of Belief Propagation on this scale is still too high for long-range correspondence estimation, a simple minima-preserving data term compression is used. During Belief Propagation, a symmetry term ensures consistent results. Occluded regions are identified and inpainted: Assuming that each occluded area is surrounded by two independently moving regions, Geodesic Matting [8] is used to propagate correspondence information. The resulting image correspondence map is upsampled to its original size and refined locally.

The approach described in this chapter has been presented in parts as a poster at SIGGRAPH 2010 [236], at the Conference for Visual Media Production (CVMP) 2010 [237], and it was an invited paper in the Journal of Virtual Reality and Broadcasting [238].

3.1 Belief Propagation for Image Correspondences

Belief Propagation [47] estimates discrete labels for every vertex in a given graph, i.e., correspondences for every pixel in a given image. Although sub-pixel accuracy is not achieved with Belief Propagation, its robustness makes it an appealing option for discrete energy minimization problems. In a nutshell, establishing pixel correspondences between two images with Belief Propagation works as follows: At each pixel location, matching costs for every possible pixel match in a given search window are computed. Typically, the L_1 norm of the pixel brightness is used for this matching cost. Neighboring pixels iteratively exchange their (normalized) matching costs. This message passing process regularizes the image correspondence problem. It finally converges to a point when consensus about per-pixel correspondences is reached. As a result, a discrete correspondence vector $\mathbf{w}_{AB}(\mathbf{x})$ is assigned to every pixel location $\mathbf{x}$ in image A that

encodes the correspondence to pixel location $\mathbf{x}' = \mathbf{x} + \mathbf{w}_{AB}(\mathbf{x})$ in image B. For a thorough introduction to Belief Propagation cf. [47].

The correspondence estimation is formulated as an energy minimization problem. The energy functional is based on the one proposed by Liu et al. [115]

$$\begin{aligned} E(\mathbf{w}_{AB}) &= \sum_{\mathbf{x}} \|d_A(\mathbf{x}) - d_B(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_1 \\ &+ \sum_{(\mathbf{x},\mathbf{y})\in\epsilon} \min(\alpha\|\mathbf{w}_{AB}(\mathbf{x}) - \mathbf{w}_{AB}(\mathbf{y})\|_1, c) \end{aligned} \tag{3.1}$$

Where $\mathbf{w}_{AB}(\mathbf{x})$ is the correspondence vector at pixel location $\mathbf{x} = (u, v)$. In contrast to the original SIFT flow implementation [115], $d_A(\mathbf{x}) = c_A(\mathbf{x}) + s_A(\mathbf{x})$ is a 131-dimensional descriptor vector, containing both color information $c_A(\mathbf{x}) \in \mathbb{R}^3$ and the SIFT descriptor $s_A(\mathbf{x}) \in \mathbb{R}^{128}$ of location $\mathbf{x}$ in image A. Each descriptor entry has a value between 0 and 255. The regularization parameters α, c are set to $\alpha = 160$ and $c = 5w$, where w is image width. In addition, the pixel neighborhood ϵ is not only defined by the image lattice (i.e., a conventional 4-neighborhood), as shown in Section 3.1.2.

As Liu et al. [115] demonstrated, this energy functional can be minimized with Efficient Belief Propagation. In the following subsections the correspondence estimation scheme is described in detail.

The correspondence estimation consists of six consecutive steps.

1. The image data is downsampled to a lower resolution using a custom SIFT descriptor downsampling scheme, cf. Section 3.1.1.

2. Custom neighborhoods are built for all pixels based on both the image lattice and appearance similarity, cf. Section 3.1.2.

3. For each pixel, all possible matches in a given search window are evaluated and compressed with a custom, minima-preserving data term compression scheme, cf. Section 3.1.3.

4. Correspondence vectors are estimated as described by Liu et al. [115], extended by a symmetric extension , cf. Section 3.1.4

5. Possibly occluded areas are inpainted, cf. Section 3.1.5.

6. Initial results are upsampled and refined locally, cf. Section 3.1.6, again making use of a custom neighborhoods, data term compression and symmetric Belief Propagation.

3.1.1 SIFT Descriptor Downsampling

Liu et al. [115] designed their SIFT flow with the goal in mind to match images that may only be remotely similar, which comes close to the original intention of Lowe et al. [119] to find only a few dominant features. My goal, on the other side, is to match very similar images. SIFT features are used to capture detail information about the scene, one feature is generated for every pixel of the full resolution images, taken from the bottom layer of the SIFT scale-space pyramid. In order to only capture the most prominent details, a single representative feature is kept for every $n \times n$ grid of pixel locations in image A.

The search for a representative descriptor is inspired by the work of Frey et al. [48] who use their Affinity Propagation technique to search for clusters in data and simultaneously identify representatives for these clusters. Since the arrangement of clusters is pre-defined, i.e., the $n \times n$ pixel block structure has to be preserved, fixed clusters are used. Their suggestion is adopted that a cluster's representative should be the one most similar to all other members of the cluster.

Hence, the representative descriptor for each pixel block is the one in the $n \times n$ pixel cell that has the lowest cumulative L_1 distance to all other descriptors. A downsampled representation of the image is computed where every grid cell is represented by a single descriptor. This descriptor consists of the mean color values of the cell and the representative SIFT descriptor.

3.1.2 Construction of Message Passing Graph

The fact that image regions of similar color often share common properties, e.g., similar motion, is often exploited in regularization techniques. Typically, this is achieved by applying an anisotropic regularization, i.e., neighboring pixels with different colors exert less influence on each other than pixels with a similar color. This technique has two drawbacks: First, regularization is decelerated, because evaluating messages with low weights takes up precious time and has little effect. Second, the grid-aligned regularization leads to jaggy borders around correspondence discontinuity edges. Recently,

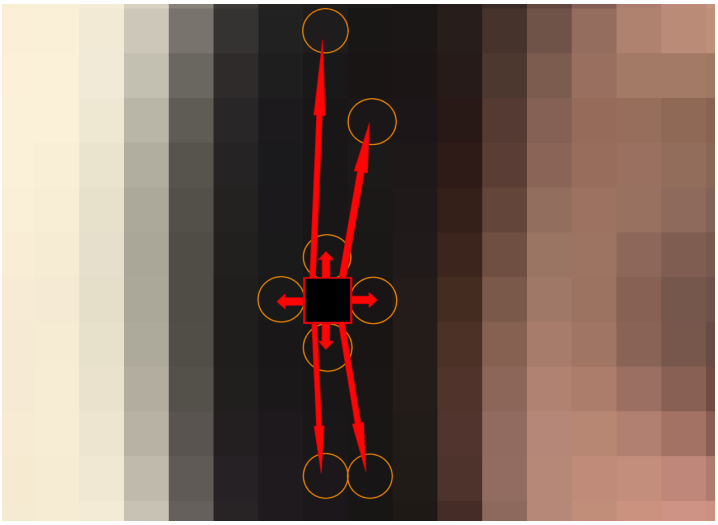
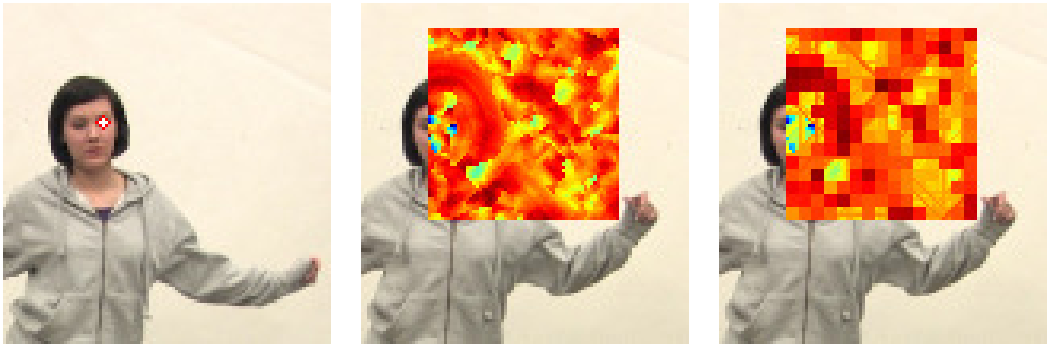


Figure 3.1: In the Belief Propagation scheme, a single pixel (red square) exchanges messages with its spatial neighbors as well as pixels of similar color (orange circles). The underlying graph structure is obtained by computing minimal spanning trees.

Smith et al. [169] proposed the construction of a non-grid-like regularization scheme. While they apply this technique to stereo computation with a variational approach, I adapt their idea to Belief Propagation, Fig. 3.1. An initial graph is built where each vertex represents a pixel location of the image. Edges connect pixels that have a certain maximal distance. Typically, this maximal distance is set to 20 pixels since pixels farther apart are rarely selected as neighbors. Each edge is assigned a weight that corresponds to the L_1 norm of the color and position of the connected pixels. As in [169], a minimum spanning tree is calculated using Kruskal's algorithm [86]. The edges of the spanning tree are stored and removed from the overall graph. Afterwards the procedure is repeated, so that the mean valence of a vertex is 4. The new found neighbors are added to the original 4-neighborhood ϵ of a pixel location.

3.1.3 Data Term Compression

One bottleneck in Belief Propagation with SIFT features is the on-the-fly evaluation of matching costs $\|s_A(\mathbf{x}) - s_B(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_1$ between pixel locations $\mathbf{x}$ in image A and $\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x})$ in image B. Liu et al. [115] precompute and store the matching costs before message passing. The alternative is to re-evaluate matching costs on demand, which happens at least once during each iteration. In this context, this results in



(a) (b) (c)

Figure 3.2: Data term compression. (a) For each pixel in a source image, matching costs have to be evaluated for Belief Propagation. (b) One common approach is to precompute matching costs in a predefined window. However, this leads to very high memory load. (c) The proposed approach uses a simple minima-preserving compression of these matching cost windows. The minima of each pixel block, each column, row and the diagonal lines of the search window are stored. During decompression, the maximum of these values determines the matching cost for a given location. While regions with high matching costs (red) are not recovered in detail, local minima are preserved with high accuracy (blue).

$262(= 131 * 2)$ memory lookups per pixel comparison. Since storing data terms is not an option with the high resolution data, and on-the-fly evaluation leads to run-times of several days, a simple data term compression technique is designed. All possible matching costs are precomputed for a single pixel $\mathbf{x}$ in A and its potential matching candidates, Fig. 3.2. Since it is likely that a pixel will be matched with a candidate having a low dissimilarity (i.e., low matching cost), a minima-preserving compression technique is employed that loses detail in areas where high matching scores prevail. For each $m \times m$ grid cell of the original data term, the minimum is stored. In addition, for each row and column of the matching window, the respective minimum is stored. The same applies to the minima along the two diagonal directions.

During decompression, the maximum of the these minima is evaluated, resulting in 5 memory lookups (the minimal grid cell value, the minimal row and column values and the minimal values of the two diagonals). By setting $m = 4$, at a data term window size of typically 160×160 pixels, the memory usage per term is reduced from $160 * 160 = 25600$ float values to $40 * 40 + 4 * 160 = 2240$ float values.

3.1.4 A Symmetric Extension

Since symmetry between bidirectional correspondence maps should be enforced, a symmetry term similar to the one proposed by Alvarez et al. [6] is introduced. To the energy functional (Eq. 3.1) a symmetry term is added:

$$\begin{aligned} E(\mathbf{w}_{AB}) &= \sum_{\mathbf{x}} \|d_A(\mathbf{x}) - d_B(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_1 \\ &+ \sum_{(\mathbf{x},\mathbf{y})\in\epsilon} \min(\alpha\|\mathbf{w}_{AB}(\mathbf{x}) - \mathbf{w}_{AB}(\mathbf{y})\|_1, c) \\ &+ \sum_{\mathbf{x}} \min(\alpha\|\mathbf{w}_{AB}(\mathbf{x}) + \mathbf{w}_{BA}(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_2, c) \end{aligned} \tag{3.2}$$

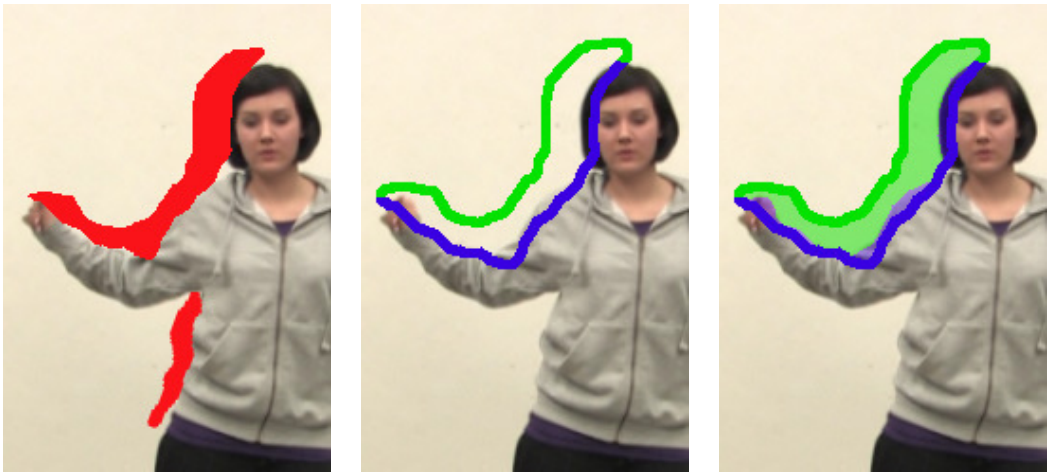
There two correspondence maps now: $\mathbf{w}_{AB}$ and $\mathbf{w}_{BA}$. They are jointly estimated and evaluated after each Belief Propagation iteration. It proved to be beneficial to assign the same weighting and truncation values α, c to the symmetry term that are also used for message propagation.

3.1.5 Occlusion Removal

It can be observed that the introduction of a symmetry term leads to symmetric warps in non-occluded areas. Hence, the symmetry $\|\mathbf{w}_{AB}(\mathbf{x}) + \mathbf{w}_{BA}(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_2$ of two opposing correspondence maps $\mathbf{w}_{AB}$ and $\mathbf{w}_{BA}$ is used as a measure of occlusion.

For each of these two simultaneously estimated maps, asymmetric correspondence regions are identified and treated independently, Fig. 3.3. First, all occluded regions are filled with correspondence information values using diffusion. All pixels which would lie outside the actual image boundaries according to their correspondence vector are treated as boundary occlusions, and their diffused values are kept.

Assuming that each of the remaining occlusion regions is confined by a foreground and a background region that move incoherently, a k-means ($k = 2$) clustering of the border region outside each occluded area is performed, all pixels in these border regions are clustered according to their two correspondence vector components of $\mathbf{w}_{AB}(\mathbf{x})$. The resulting pixel sets serve as input data to a binary Geodesic Matting [8] that assigns each pixel in the occluded area a foreground or background label. After the labeling is computed, the median value of the n nearest neighbors of the foreground or background region is assigned. Typically, n is set to $n = 20$ to get smooth results.



(a) (b) (c)

Figure 3.3: Occlusion removal. (a) Regions with asymmetric correspondences (red) are processed in a two-step algorithm. (b) First, a k-means clustering (k=2) reveals the two predominant offset directions (blue and green). (c) These two image regions are used as input for Geodesic Matting. Depending on which label is assigned to an occluded pixel, the local median foreground or background motion is assigned.

3.1.6 Upsampling and Refinement

The low resolution correspondence is upsampled as follows. On the high-resolution map, each pixel that was chosen as the SIFT descriptor representative is assigned the values that result from the low-resolution Belief Propagation (scaled by factor n). For all remaining pixels, the value of the nearest representative pixel in (image brightness) gradient space is assigned. These assigned correspondence values serve as a prior for a local refinement. Like on the low resolution level, symmetric Belief Propagation is used to obtain the final per-pixel correspondence. The crucial difference is that the search window is set to a very small size (typically $(n * 2 + 1) \times (n * 2 + 1)$ pixels, where n is the original downsampling factor) and that it is located around a correspondence prior $\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x})$ and not around the pixel location $\mathbf{x}$ itself.

3.2 Results and Discussion

In order to showcase the strengths of my approach, the estimated correspondence fields are applied to pairwise image morphing. While traditional approaches usually employ

a user-assisted workflow [16], the presented approach strives to compute motion vectors between images automatically. A simple forward warping scheme is used to seamlessly render intermediate views between two frames.

A GPU rendering approach, inspired by Stich et al. [177], is used. A dense vertex mesh for each input image is created and forward warped according to the high-resolution flow field. Fragments are discarded whose local divergence in the correspondence map exceeds a threshold of 4 pixels. If ambiguities arise (i.e., two fragments of a mesh overlap), the fragment with lower symmetry is discarded. The two forward-warped meshes are alpha-blended. Pixel fragments with very low symmetry are again discarded in the presence of pixels with symmetric correspondence. Fig. 3.4 shows some image warping results.

For all following scenarios, image pairs with both camera and object motion were deliberately chosen. The Breakdancer scene taken from the dataset of Zitnick et al. [211] features noisy images and fast scene motion. Still, image correspondences are successfully established. As seen in Fig. 3.4 (c, top), the shadow of the breakdancer in the background moves plausibly. The shortcomings of the simple rendering approach manifest themself in motion streak artifacts around the right foot of the breakdancer. Note that in their original work, Zitnick et al. [211] only performed stereo matching. The presented approach does not exploit the epipolar constraint and copes with moving objects and shadows. The dancer scene looks a lot less challenging at first glance. However, the ground surface is quite demanding, since shadows and reflections of the dancer and background are visible. The Parkour scene can be interpreted as a failure case of my approach. Although large parts of the scene are matched correctly, the occluded regions around the Parkour runner are not handled correctly by the Geodesic Matting. The background is too cluttered to allow for a consistent local color model that separates background from foreground. The Fireball sequence shows that the algorithm copes well with illumination changes. However, the opening crack around the Fireball impairs overall rendering quality.

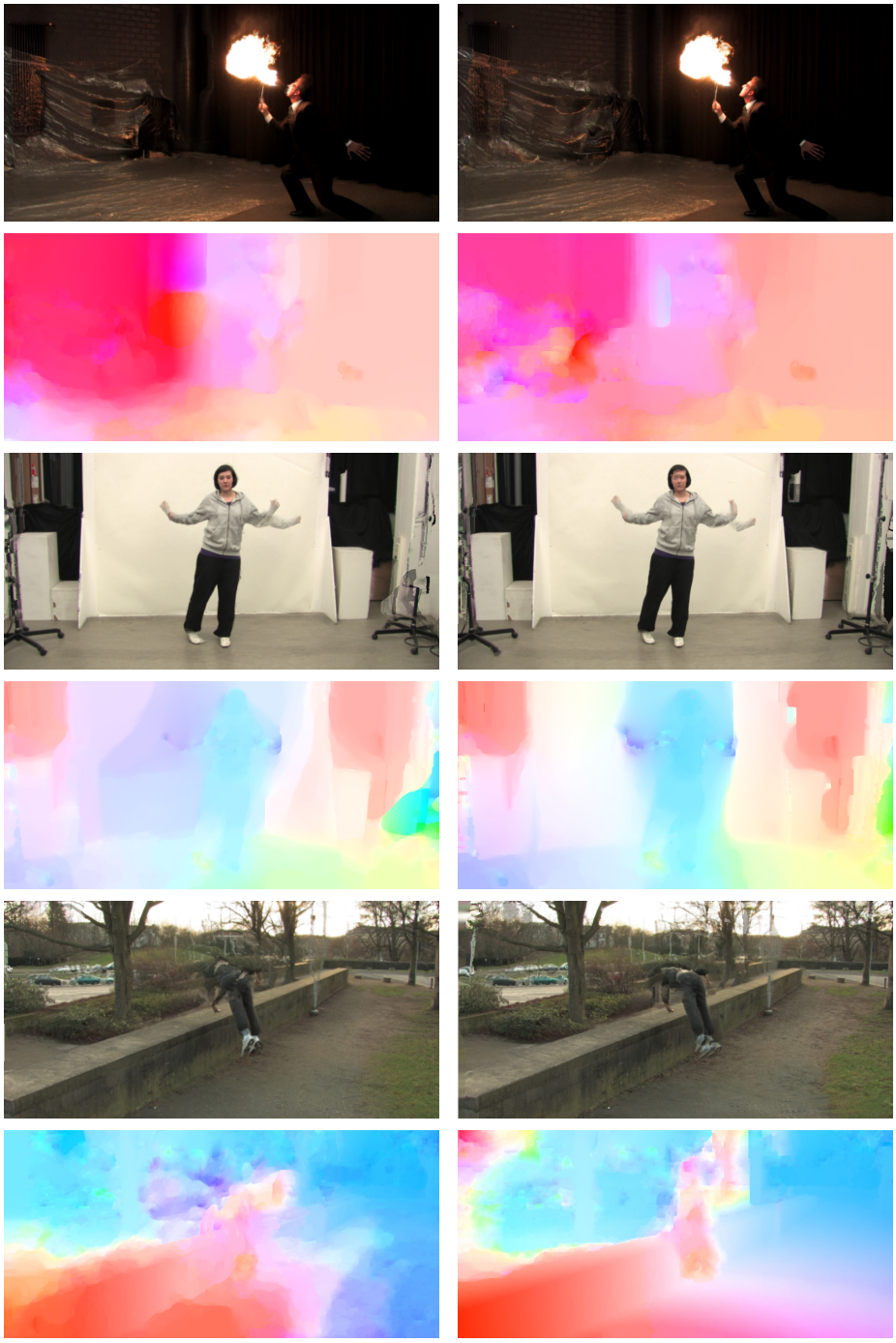
To emphasize the high complexity of the test scenes, optical flows for the Parkour, Dancing, and the Fireball scene were also estimated with three state-of-the-art optical flow implementations of Werlberger [197], Brox [23] and Pock [31]. Figure 3.5 shows the flow fields and interpolated images side-by-side in comparison to my approach. Notice that the same rendering technique is used to generate all the images. The fast

(a) (b)

Figure 3.4: Image morphing results: Input images.

(c) (d)

Figure 3.4: (cont'd) (c) offset vector fields from first to second image, coded in optical flow notation, (d) rendered in-between image. Note that while robust results are obtained, details such as the reflection of the dancer or the shadow of the breakdancer are preserved.



(a) (b)

Figure 3.5: Comparison of image interpolation results using state-of-the-art optical flow algorithms. From top to bottom, the columns of the figure show an interpolated image and below the color-coded correspondence field. Errors in the estimated flow field typically show up as ghosting artifacts. Results are shown in columns, computed using (a) the GPU implementation by Werlberger et al. [197], (b) TV-L^1 motion estimation by Chambolle and Pock [31]

(c) (d)

Figure 3.5: (c) large displacement optical flow by Brox and Malik [23], and (d) the proposed approach. Only my approach is able to correctly estimate the motion of the background in the fire scene (row 1 and 2) and the large arm motion of the dancer (row 3 and 4). In the Parkour scene (row 5 and 6), my approach fails at the borders of the dancer, but correctly estimates correspondences of the trees in the background and manages to find the huge displacement of the wall due to change of camera perspective.

GPU implementation of Werlberger et al. [197] focuses on quick runtime and allows for computing the image correspondences within several seconds. However, the rendered results using this approach show severe visual artifacts. When comparing to the TV-L^1 motion estimation by Chambolle and Pock [31] and the large displacement optical flow by Brox and Malik [23], it becomes apparent that the proposed technique bears great potential. While the actual foreground objects are covered well by all algorithms, challenging details in the background, e.g., the trees (Parkour scene) or the plastic foil (Fireball scene), are correctly matched only by my approach.

3.2.1 Limitations

The most severe limitation is the long run-time which only allows processing single image pairs or short video clips. Until now, the implementation is completely CPU based. This is also due to high memory consumption. Since two symmetric correspondence maps are computed in parallel, memory consumption is as high as 4 GB for large displacements (e.g., the Parkour sequence which features displacements of up to 400 pixels). In order to move the computation to GPU, an even more compact data representation than the currently employed compression scheme has to be developed. Obtaining a suitable search window size is another problem, since the maximum displacement has to be known prior to the correspondence estimation. If the window size is set too small, some correspondences will not be established correctly. If it is set too high, the overall run-time will be excessively high.

4

Free Viewpoint Video using Multi-Image Warping

In this chapter I will describe how pairwise image correspondences can be used to realize free viewpoint renderings of arbitrary real-world scenes. I strive to be able to work with "casually" captured scenes, i.e., material that was captured with off-the-shelf, unsynchronized cameras in arbitrary indoor and outdoor environments. In general, it is very hard to faithfully reconstruct the geometrical properties when cameras are not synchronized, precisely calibrated, or the scene content is arbitrary. Therefore, I employ image warping techniques for rendering novel viewpoints, circumventing the necessity of exact depth or geometry reconstruction, cf. Section 2.2.3.

In Section 4.1 I recapitulate the basic rendering equation for multi-image interpolation and describe the required steps for extending this rendering scheme to free viewpoint video in Section 4.2. The actual rendering equation along with some implementation details follows in Section 4.3. Possibilities to create stereoscopic output material are described and discussed in Section 4.4. Practical aspects, i.e., pre-processing and authoring of free viewpoint data are tackled in Section 4.5, before results are evaluated in Section 4.6. The integration into an actual video production pipeline is documented in Section 4.7.

The algorithms presented in this chapter have been published in part as a SIGGRAPH 2009 poster [234], in Computer Graphics Forum (CGF) [235] and were presented at Eurographics 2011. Note that the research on multi-image rendering [226] and the space-time visual effects [228] by Linz et al. is based on the *Virtual Video*

Camera rendering pipeline. Therefore, a short summary of the rendering procedure (Sections 4.2, 4.5, and 4.3, including Figs. 4.1 and 4.9) can be found in his dissertation [98]. Timo Stich also documented the early research on spatio-temporal free viewpoint rendering: A sketch of the space-time renderer can be found in his dissertation [176], albeit without the neccessary concepts of space-time embedding and constrained Delaunay tesselation.

4.1 (Multi-)Image Warping

Let us recapitulate the general correspondence-based rendering equation. For a given pixel location $\mathbf{x}$ in an image A we know the corresponding pixel position $\mathbf{x}' = \mathbf{x} + \mathbf{w}_{AB}(\mathbf{x})$ in an image B. Its pixel position can be moved towards its corresponding counterpart according to a weight b.

$$\tilde{I}_A(\mathbf{x} + b\,\mathbf{w}_{AB}(\mathbf{x})) = I_A(\mathbf{x}) \tag{4.1}$$

where $\tilde{I}_A$ is the forward warped representation of image I_A: for each original pixel location, an updated position is determined. Similarly, we can create a warped version of I_B and blend the two warped images according to the interpolation weights a, b:

$$I_V(\mathbf{x}) = a\,\tilde{I}_A(\mathbf{x}) + b\,\tilde{I}_B(\mathbf{x}) \tag{4.2}$$

More than two images can be used for this warping scheme [88]. Instead of only two weighting parameters $\{a, b\}, a + b = 1$ an arbitrary number of weighting parameters $\{\mu_1, ..., \mu_n\}, \sum_i \mu_i = 1$ can be used. Chen and Williams proposed to apply this kind of rendering for spatial view interpolation [33] and demonstrated results for synthetic scenes and regular, grid-like camera setups. I take their approach one step further and demonstrate the feasibility of correspondence-based rendering for real-world scenes, arbitrary camera placements and joint spatio-temporal view synthesis.

4.2 From Image Warping to Free Viewpoint Video

Although it is theoretically possible to let all available source images contribute to the novel viewpoint synthesis, it is not feasible to do so. Instead, only a handful of spatial and temporal neighbors are used to synthesize a novel view of the scene.

A three-dimensional spatio-temporal navigation space $\mathcal{N}$ contains all captured images as well as any desired virtual view. For each image, the rotation and translation of the capturing camera as well as the recording time are obtained. Using a mapping function Ψ, the navigation space parameters are determined for all captured images. The navigation space is partitioned, so that each novel view is enclosed in exactly one tetrahedron that is defined by four original images. The warping and blending weights needed for rendering the novel view correspond to the barycentric weights of this particular tetrahedron. In the next paragraphs, this procedure is explained in detail.

Camera Calibration. Rotation $\mathbf{R}$ and projection center $\mathbf{p}$ of each camera are needed to define the mapping Ψ from world space coordinates to navigation space $\mathcal{N}$. Recent structure-from-motion algorithms for unordered image collections [60] solve this problem robustly and can also provide a set of sparse world space points needed for constructing the common ground plane for the navigation space. This algorithm yields robust results also for dynamic scenes. For dynamic camera scenarios (i.e., handheld, moving cameras), $\mathbf{R}$ and $\mathbf{p}$ have to be computed for every frame of each camera.

Temporal Registration. The mapping Ψ additionally needs the exact recording time t of each camera. The sub-frame temporal offset is estimated by recording a dual-tone sequence during acquisition and analyzing the audio tracks afterwards [69]. If recording a separate audio track is not feasible, pure image-based approaches [130] can be employed instead. Note that after temporal registration, the recorded images are still not synchronized. Since the camcorder recordings are triggered manually, a sub-frame offset still prevents exploiting the epipolar constraint for moving objects.

Navigation Space Embedding. My goal is to explore the captured scene in an intuitive way and render an image I_V of a virtual view V, corresponding to a combination of viewing direction and time. To this end, I choose to define a 3-dimensional navigation space $\mathcal{N}$ that represents spatial camera coordinates as well as the temporal dimension. In their seminal paper, Chen and Williams [33] propose to interpolate the camera rotation $\mathbf{R}$ and position $\mathbf{p}$ directly in 6-dimensional hyperspace. While this is perfectly feasible in theory, it has several major drawbacks in practice: it neither allows for intuitive exploration of the scene by a user, nor is it practical to handle the amount

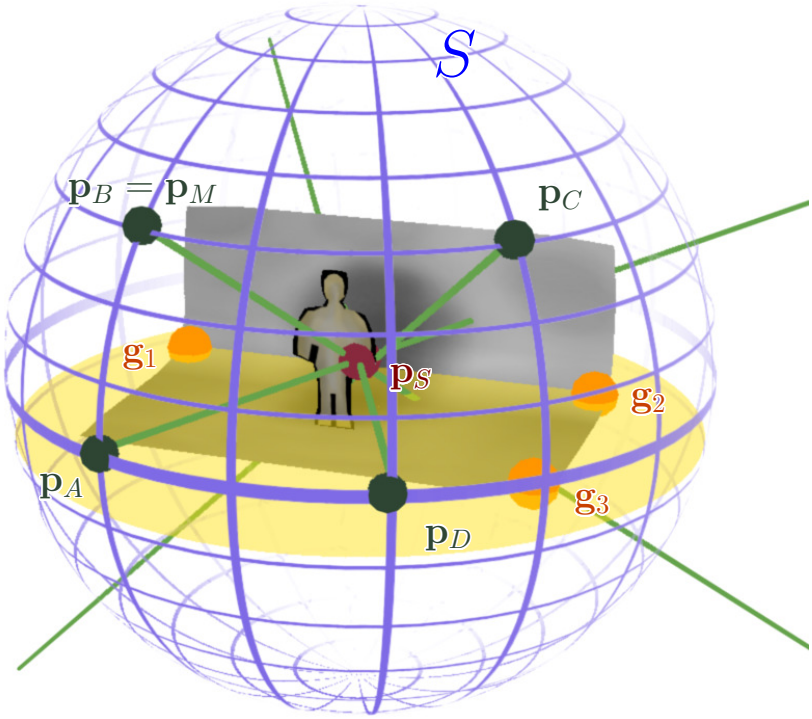


Figure 4.1: Navigation space: A sphere S is defined around the scene. For the center $\mathbf{p}_S$ of S, the point closest to the optical axes of all cameras (green lines) is least-squares-determined. The user selects three points $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ to define the ground plane (yellow circle). The normal of the plane as used as the up vector of the scene, and thus as the rotation axis of the sphere. The embedding is uniquely defined by labeling one of the cameras $\{\mathbf{p}_A, ..., \mathbf{p}_D\}$ as the master camera $\mathbf{p}_M$.

of emerging data needed for interpolation in this high-dimensional space. Additionally, cameras would have to be arranged in such a way as to span an actual volume in Euclidean space. With this requirement, it would be hard to devise an arrangement of cameras where they do not occlude each other's view of the scene. Chen and Williams exploit the fact that their synthetic scenes allow to use a subspace of $\{\mathbf{R}, \mathbf{p}\}$: Since they can exactly control the camera parameters for rendering input images, they can easily produce images that reside on a two-dimensional subspace, e.g., horizontal and vertical movement of the camera center. With real-world footage this is hardly possible, especially with "casually" captured data sets. The crucial design decision in the proposed system is hence to map the extrinsic camera parameters to a lower dimensional space that allows intuitive navigation. The temporal dimension already defines one axis of

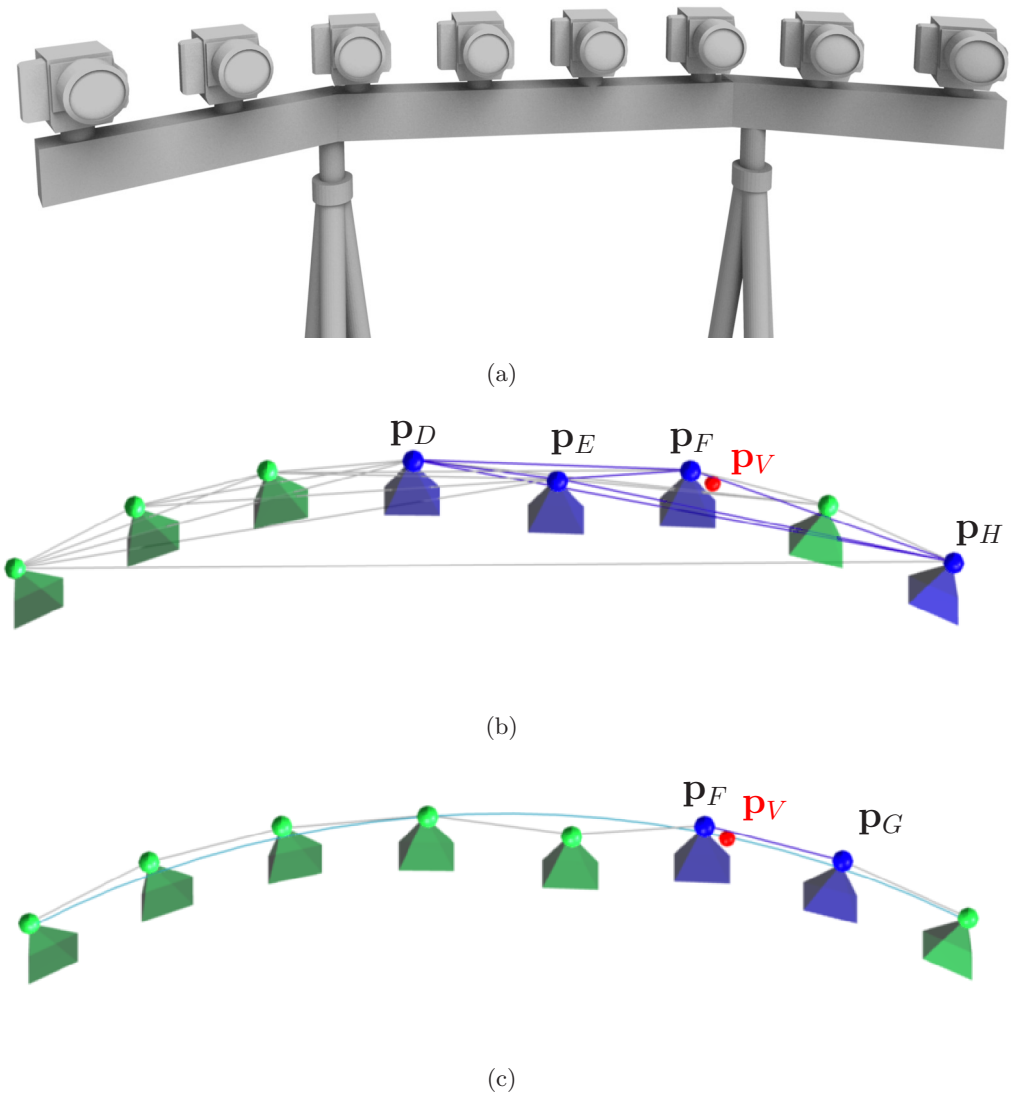


Figure 4.2: Two possibilities to partition the camera setup of the *Breakdancer* sequence. (a) depicts the actual camera setup (camera placement reproduced from [211]). When partitioning the Euclidean space directly (b), several problems arise. Although an actual volume is spanned by the cameras, many tetrahedra are degenerated. If the virtual camera V with projection center $\mathbf{p}_V$ is reconstructed, captured images from cameras $\mathbf{p}_D$, $\mathbf{p}_E$, $\mathbf{p}_F$ and $\mathbf{p}_H$ as well as the wide-baseline correspondence fields between them are required. Using the navigation space embedding (c), interpolation only takes place between neighboring cameras. Less data has to be processed and correspondence estimation is much easier. Additionally, spatial navigation is intuitively simplified. Instead of a 3D position, only the position on the one-dimensional arc (light blue) has to be specified. The small spatial inaccuracy (distance between $\mathbf{p}_V$ and line segment between $\mathbf{p}_F$ and $\mathbf{p}_G$) is negligible even for ad-hoc setups.

the navigation space $\mathcal{N}$, leaving two dimensions for parametrizing the camera orientation and position. Practical parametrizations that allow for realistic view interpolation are only possible if the cameras' optical axes cross at some point (possibly at infinity).

A natural choice for such an embedding is a spherical parametrization of the camera setup, Fig. 4.1. While, for example, a cylindrical embedding or an embedding in a plane is also feasible, a spherical embedding allows for all reasonable physical camera setups, ranging from cameras arranged in a 1-dimensional arc, over cameras placed in a spherical setup to linear camera arrays with parallel optical axes in the limit. Regarding existing data sets, it is obvious that spherical/arc-shaped setups are the predominant multi-view capture scenarios [149, 211, 42, 38, 157, 37]. Even in unordered image collections it can be observed that a majority of camera positions is distributed in arcs around certain points of interest [172]. Other placements, such as panoramic views, are also possible, but would suffer from the small overlap of the image regions of cameras. For all results presented in this paper, a spherical model with optical axes centered at a common point is employed. Camera setups such as Bézier splines or patches are also feasible and the presented approach may adapt to any setup as long as a sensible parametrization can be obtained. As the extension to these settings is straightforward, it will not discussed it in detail.

(Virtual) cameras are placed on the tesselated surface of the virtual sphere S, Fig. 4.3 (right). Their orientations are defined by azimuth φ and elevation θ. Together with the temporal dimension t, φ and θ span the 3-dimensional navigation space $\mathcal{N}$. If cameras are arranged in an arc or curve around the scene, θ is fixed, reducing $\mathcal{N}$ to two dimensions. As this simplification is trivial, only the 3-dimensional case is covered in the following discussion. In contrast to the conventional partition of space suggested by Chen and Williams [33], the movement of the virtual camera is restricted to a subspace of lesser dimensionality (2D approximate spherical surface or 1D approximated arc). Although this might appear as a drawback at first sight, several advantages arise from this crucial design decision, Fig. 4.2:

1. The amount of correspondence fields needed for image interpolation is reduced significantly, making both pre-processing and rendering faster.

2. An unrestricted partition of Euclidean space leads to degenerated tetrahedra. Especially when cameras are arranged along a line or arc, adjacencies between

remote cameras are established for which no reliable correspondence information can be obtained.

3. The parametrization of the camera arrangement provides an intuitive navigation around the scene.

To define navigation space $\mathcal{N}$, ground-truth extrinsic camera parameters $\mathbf{R}$ and $\mathbf{p}$ are assumed for every camera, as well as a few point correspondences with their 3D world coordinates. For a specific virtual image $I_V(\varphi, \theta, t)$, the desired output of the free viewpoint renderer is a (virtual camera) image at a given point in navigation space defined by the two spatial parameters (θ, ϕ) as well as recording time t. To serve as sampling points, the camera configuration of the recorded multi-video input in Euclidean world space is embedded into navigation space $\mathcal{N}$

$$\Psi : (\mathbf{R}, \mathbf{p}, t) \mapsto (\varphi, \theta, t).$$

The function Ψ is simply a transformation from Cartesian coordinates to spherical coordinates, where the sphere center $\mathbf{p}_S$ and the radius of the sphere r_S are computed from the cameras' extrinsic parameters $\mathbf{R}$ and $\mathbf{p}$ in a least-squares sense.

The embedding is uniquely defined by specifying a ground plane in the scene and by labeling one of the cameras $\{\mathbf{p}_A, ..., \mathbf{p}_D\}$ as the master camera $\mathbf{p}_M$, Fig. 4.1.
It is evident that the recording hull spanned by all camera positions is, in general, only a crude approximation to a spherical surface. However, non-spherical camera arrangements do not cause any visually noticeable effect during rendering.

4.2.1 Spacetime tetrahedralization

In order to interpolate viewpoints, Chen and Williams [33] propose to generate some arbitrary graph to partition the space such that every possible viewpoint lies in a d-simplex and can be expressed as a linear combination of the $d + 1$ vertices of the enclosing simplex. When including the temporal dimension, however, an arbitrary graph leads to rendering artifacts due to inconsistent mappings between Euclidean space and navigation space. Let me illustrate this effect with an example:

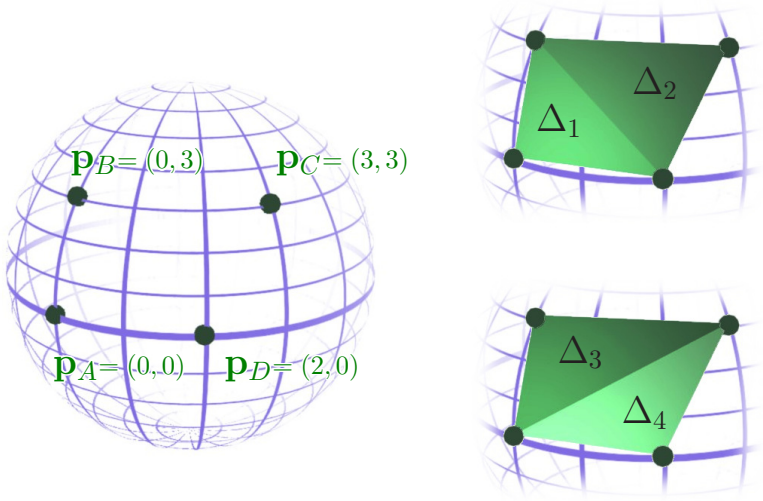


Figure 4.3: Surface tessellation: for the spherical coordinates of four cameras $\mathbf{p}_A, \ldots \mathbf{p}_D$, two possible tessellations $\{\Delta_1, \Delta_2\}$ and $\{\Delta_3, \Delta_4\}$ exist. During space-time interpolation, the configuration of the initial tessellation may not change to avoid interpolation discontinuities.

Naïve Tetrahedralization. The navigation space consists of two view-directional and the temporal dimension. The navigation space is subdivided into tetrahedra τ such that each embedded video frame represents one vertex [229]. Any virtual camera view is interpolated from the enclosing tetrahedron $\tau = \{\mathbf{v}_i\}, i = 1 \ldots 4$. As already mentioned, arbitrary partitions are feasible in static scenes. However, the spherical approximation and/or calibration imprecision will lead to interpolation errors in practice if the temporal dimension is included.

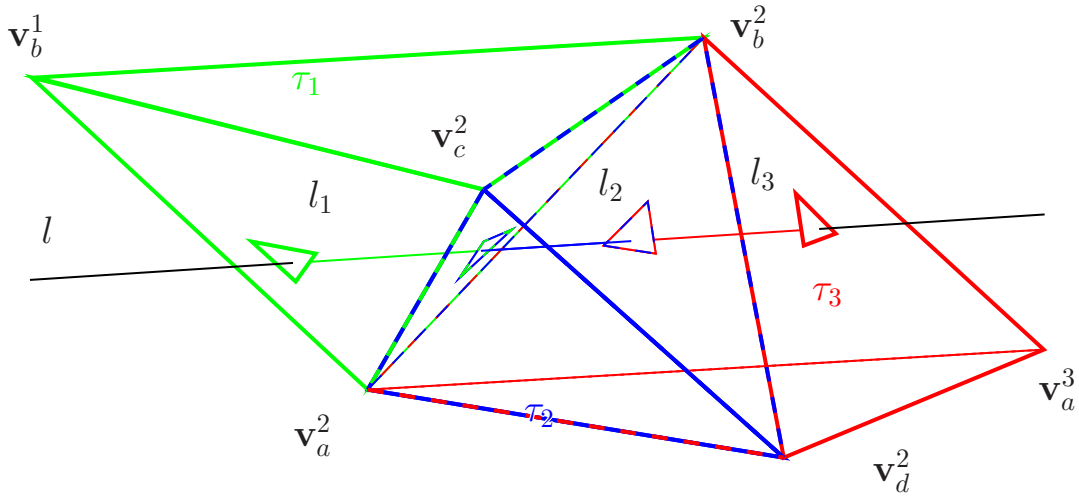
To prove this point, let us assume that we have captured a dynamic scene with four static, unsynchronized cameras with projection centers $\mathbf{p}_A, \mathbf{p}_B, \mathbf{p}_C$ and $\mathbf{p}_D$. The view-directional subspace relates to the spherical camera placement. To be more exact, it represents an approximation of a spherical surface. Examples for possible approximations are given in Fig. 4.3, upper right and lower right. I will show that although the chosen spherical approximation is arbitrary, it is crucial to enforce consistency of the approximation when the temporal dimension is introduced, even if ground-truth camera parameters $\mathbf{R}$ and $\mathbf{p}$ are available.

Imagine now that the user intends to re-render the scene from a *static* virtual camera. The position of this camera is somewhere between the four original cameras, so

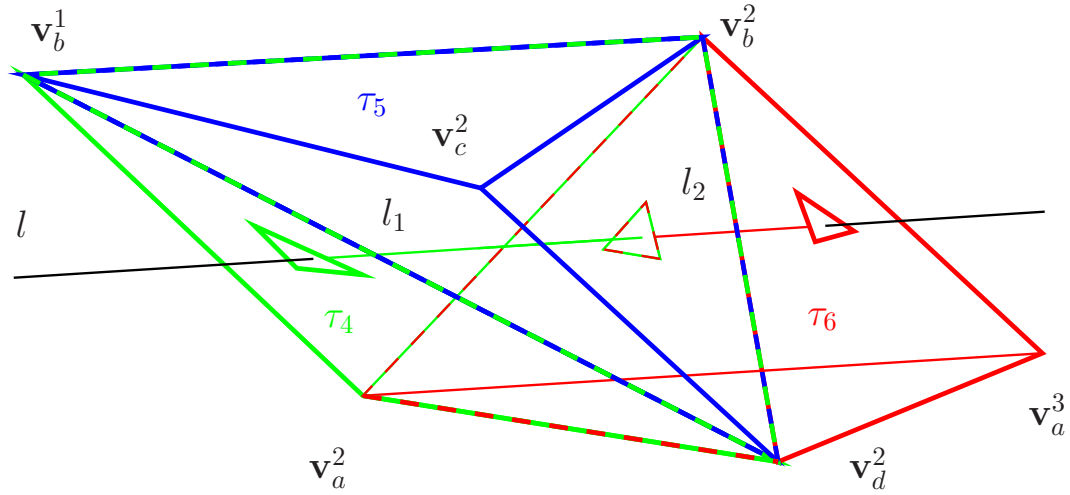
that the new view has to be interpolated. Because the camera is static in navigation space, φ and θ stay fixed while t is variable such that the virtual camera moves along a line l in navigation space, Fig. 4.4. Assume that the navigation space has been partitioned by applying a standard Delaunay tetrahedralization [41] to the set of navigation space vertices $\mathbf{v}_a^i, \mathbf{v}_b^i, \mathbf{v}_c^i, \mathbf{v}_d^i$, where i indexes the temporal dimension. Now consider the camera path l. In line section l_1 (green), images associated with camera (centers) $\mathbf{p}_A, \mathbf{p}_B$ and $\mathbf{p}_C$ are used for interpolation. The virtual camera thus represents a point that lies on plane Δ_3 in the original Euclidean space, Fig. 4.3 (lower right). In line section l_2 (blue), however, the virtual camera is interpolated from all four cameras, i.e., it *moves* within the volume spanned by the four different cameras in Euclidean world space, Fig. 4.4(a). This violates the notion of a static virtual camera. In line segment l_3 (red), the virtual camera is represented by a point in Euclidean world space situated on plane Δ_1, Fig. 4.3 (upper right). Since Δ_1 and Δ_3 are not coplanar, the position of the virtual camera in Euclidean space is different in both line sections, again violating the static camera assumption.

In navigation-space interpolation, this error can be avoided if any two points $\mathbf{v}^t = (\varphi, \theta, t)$ and $\mathbf{v}^{t+\delta} = (\varphi, \theta, t + \delta)$ re-project to the same point in Euclidean space, i.e., if the virtual camera is static in Euclidean space. Instead of building an arbitrary graph structure on the navigation space vertices, a combination of constrained and unconstrained Delaunay tessellations is applied to navigation space vertices.

A Constrained Tetrahedralization Algorithm. To obtain a temporally consistent tetrahedralization, an unconstrained Delaunay tessellation is applied on the (spatial) camera manifold, i.e., the temporal dimension is neglected for now. This initial tessellation must be preserved, i.e., the spatial ordering of cameras must stay the same for all recorded frames of the video streams. Referring back to the example given above, the tessellation shown in Fig. 4.5 (left) is obtained. The edges of this tessellation serve as boundaries that help to maintain the initial tessellation. The temporal dimension is then added. The initial tessellation of the sphere is duplicated and displaced along the temporal axis. In order to maintain the structure of the tessellation computed so far, boundary faces are added for each edge of the initial tessellation, e.g., a face $(\mathbf{v}_b^1, \mathbf{v}_b^2, \mathbf{v}_d^2)$ is added in the example above, Fig. 4.5 (right).



(a) Unconstrained Delaunay tetrahedralization.



(b) Constrained Delaunay tetrahedralization.

Figure 4.4: Unconstrained vs. constrained tessellation of the navigation space: with an unconstrained tetrahedralization (a), tetrahedra τ_1, τ_2 and τ_3 are created. A *static* virtual camera moves along line l and passes through all three tetrahedra. When comparing the line sections l_1 (green), l_2 (blue) and l_3 (red), the point in navigation space relates to different points in Euclidean space, thus the virtual camera seems to *move*. By introducing boundary faces, such errors can be avoided (b). The virtual camera passes solely through tetrahedra (τ_4, τ_6) which are spanned by the same three cameras and is not influenced by $\mathbf{v}_c^2$ (tetrahedron τ_5 remains obsolete). Therefore, the virtual camera remains static in Euclidean space, as intended.

A constrained Delaunay tessellation can then be computed on this 3-dimensional point cloud in navigation space [168]. In this example, the added boundary faces separate the data in such a way that every tetrahedron corresponds to one of the two planes in Euclidean space, i.e., either to Δ_1 or Δ_2 in Fig. 4.3, but never to Δ_3 or Δ_4. All tetrahedra also comprise of four images that were captured by exactly three cameras. If each image vertex in a tetrahedron was captured by a different camera, the situation shown in line section l_2 (Fig. 4.4(a)) would arise.

The undesired flipping along the temporal dimension also occurs when working with dynamic cameras. Here one can assume that the cameras move, but their relative position do not change, i.e., cameras cannot be allowed to switch position. In this case, the same reasoning as presented above for static cameras applies. However, now the notion of a static virtual camera is defined with respect to the moving input cameras. By doing so, sudden changes of the spherical surface approximation can be prevented, just as in the case of stationary recordings. The only limitation is that, when cameras move in Euclidean space, the approximated spherical surface also changes. In these cases the approximation error, i.e., the distance of the rendered virtual camera to the idealized spherical camera arrangement, does not remain fixed. In general, this is only noticeable with fast moving cameras and can usually be neglected for small camera movements. The same applies to imprecise camera calibration. Small errors in $\mathbf{R}$ and $\mathbf{p}$ are inevitable, but usually do not manifest in visible artifacts. In cases where the scene center is actually moving (e.g., at outdoor sports events) and cameramen are following it, a time-dependent camera embedding should be employed.

4.3 Correspondence-Based Rendering (CBR)

Having subdivided navigation space $\mathcal{N}$ into tetrahedra, each point $\mathbf{v}$ is defined by the vertices of the enclosing tetrahedron $\tau = \{\mathbf{v}_i\}, i = 1 \dots 4$. Its position can be uniquely expressed as $\mathbf{v} = \sum_{i=1}^{4} \mu_i \mathbf{v}_i$, where μ_i are the barycentric coordinates of $\mathbf{v}$. Each of the 4 vertices $\mathbf{v}_i$ of the tetrahedron corresponds to a recorded image I_i. Each of the 12 edges e_{ij} correspond to a correspondence map $\mathbf{w}_{ij}$, that defines a translation of a pixel location $\mathbf{x}$ on the image plane. A novel image $I_V(\varphi, \theta, t)$ can be synthesized for every point $\mathbf{v}$ inside the recording hull of the navigation space $\mathcal{N}$ by multi-image

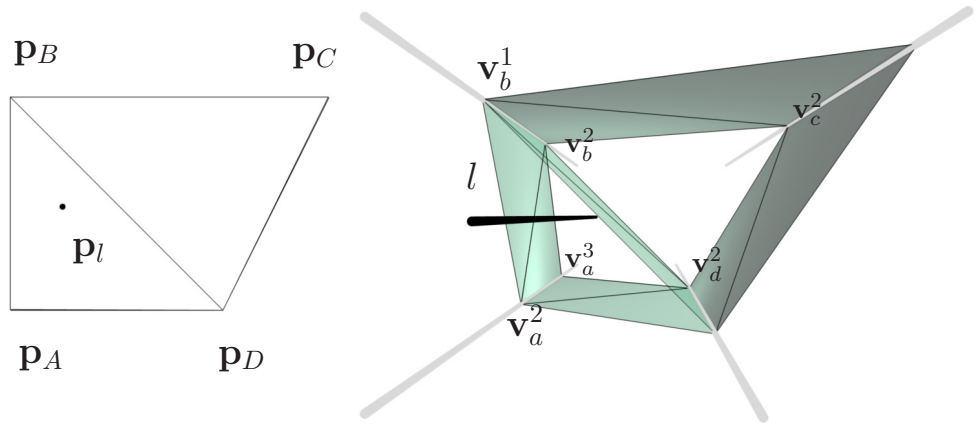


Figure 4.5: Boundary faces (green): A two-dimensional unconstrained Delaunay tessellation reveals the boundary edges between cameras (left). In 3-dimensional navigation space (right), boundary faces (green triangles) are inserted for each boundary edge. The virtual static camera V_l that moves on a line l in navigation space always lies in tetrahedra constructed from camera projection centers $\mathbf{p}_A$, $\mathbf{p}_B$ and $\mathbf{p}_D$, as postulated in Fig. 4.3.

interpolation:

$$I_V(\varphi, \theta, t) = \sum_{i=1}^{4} \mu_i \tilde{I}_i, \tag{4.3}$$

where

$$\tilde{I}_i \left(\Pi_i(\mathbf{x}) + \sum_{j=1,\ldots,4, j\neq i} \mu_j \Pi_j(\mathbf{w}_{AB}(\mathbf{x})) \right) = I_i(\mathbf{x}) \tag{4.4}$$

are the forward-warped images [124]. $\{\Pi_i\}$ are a set of re-projection matrices that map each image I_i onto the image plane of $I_V(\varphi, \theta, t)$, as proposed by Seitz and Dyer [159]. Those matrices can be derived from camera calibration. For a thorough description I would like to refer to the dissertation of Christian Linz [98]. Since the virtual image $I_V(\varphi, \theta, t)$ is always oriented towards the center of the scene, this re-projection corrects the skew of optical axes potentially introduced by the loose camera setup and also accounts for jittering introduced by dynamic cameras. Image re-projection is done on the GPU without image data resampling. Occlusion/disocclusion are handled on-the-fly based on correspondence field heuristics as proposed by Stich et al. [179]. Given the images recorded by two neighboring cameras at roughly the same point in time, the horizontal component of the correspondence fields leading from the left camera image to the right camera image is evaluated. Assuming that inter-camera motion is generally larger than inter-frame motion, a simple visibility ordering heuristic is used. An object

located at the center of the scene will exhibit a horizontal motion vector close to zero, objects in the foreground move to the right and objects in the back move to the left. When two regions of a warped image overlap, i.e., two pixel fragments are warped onto the same location, a simple comparison of their respective motion vectors thus resolves this ambiguity.

Disocclusions are detected by calculating local divergence in the correspondence fields. If any two neighboring pixels exhibit a difference of more than 4 pixels in their motion, triangles connecting them are discarded in a geometry shader. Hole filling is done during the blending stage where image information from the other three warped images is used. As a last step, the borders of the rendered images are cropped (10% of the image in each dimension), since no reliable correspondence information is available for these regions.

4.4 Stereoscopic Rendering

Before I dive into the details of the free viewpoint rendering approach, I would like to evaluate the possibilities to create stereoscopic content from within a free viewpoint system. Although the free viewpoint renderer described in the previous sections is used as a foundation, the following discussion is applicable to most image-based free viewpoint video approaches.

Since it is important to understand that all required spatial information for stereoscopic rendering is already implicitly available, I will investigate the correspondence fields used in correspondence-based rendering. I then present two possible ways to render stereoscopic video and juxtapose their benefits.

4.4.1 Correspondence Fields

To be able to create image-based stereoscopic images, it is important to get an insight into the nature of the information contained in the correspondence fields. A correspondence field is a dense vector field $\mathbf{w}_{ij}$ directed from source image I_i to a destination image I_j. One option of creating those correspondence maps is the application of optical flow algorithms to the source and destination image.

In order to get a clear understanding of the information encoded within $\mathbf{w}_{ij}$, a two-camera setup is assumed. Fig. 4.6 shows such a simplified setup where the cameras

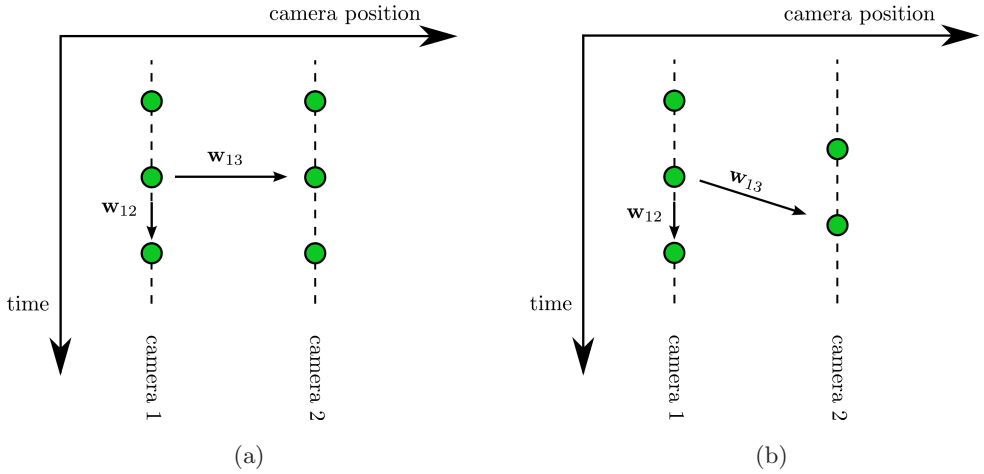


Figure 4.6: The layout of images shown as green dots in the space-time plane with different camera configurations: (a) synchronized cameras (b) unsynchronized cameras

are restricted to a 1D movement along the horizontal axis. The vertical axis is time and each green dot corresponds to an image acquired at that specific time and place. Dotted lines connect images taken by the same camera. In the following, I investigate the information contained in the correspondence fields $\mathbf{w}_{12}$ and $\mathbf{w}_{13}$ when different constraints are posed on the recording modalities.

I start with the most restrictive camera setup of static cameras and synchronized camera shutters as, shown in Fig. 4.6(a). The images are acquired on an axis-aligned regular grid in the space-time plane. The correspondence field within the first camera $\mathbf{w}_{12}$ links two consecutive images of the video stream. Since the viewpoint does not change, all image changes encoded within $\mathbf{w}_{12}$ represent motion of objects within the scene. In contrast, the correspondence field $\mathbf{w}_{13}$ linking two adjacent cameras contains no object movement at all. Source and destination image have been acquired at the same point in time. All changes along $\mathbf{w}_{13}$ can be explained by the motion parallax due to the changed viewpoint between cameras. If the source and destination image of $\mathbf{w}_{13}$ are rectified, the correspondence field amounts to a disparity map.

When the recording constraints are relaxed to a setup where the camera shutters are no longer synchronized, the interpretation of $\mathbf{w}_{13}$ changes. While the intra-camera correspondence field $\mathbf{w}_{12}$ still encodes only object motion, the inter-camera correspondence field $\mathbf{w}_{13}$ is no longer horizontally aligned with the time axis. In the space-time plane, this amounts to a sheering of the two camera paths as depicted in Fig. 4.6(b). As

a result of the unsynchronized shutters, the objects in a dynamic scene do not obey the epipolar constraint: The variation between source and destination image is no longer only defined by the viewpoint change.

Inherent to this model is the restriction to linear motion. This affects the object motion within the scene because a correspondence field defines a line along which each point can move in the image plane.

4.4.2 Direct Stereoscopic Virtual View Synthesis

According to the correspondence-based rendering scheme (Section 4.3), all recorded images are embedded into a three-dimensional navigation space $\mathcal{N}$, ands a constrained Delaunay tessellation is performed.

Instead of rendering a single output image, a left- and a right-eye view of the scene are synthesized. Typically, the initial synthetic image is used as the output for the left eye $I_V^L(\varphi, \theta, t) = I_V(\varphi, \theta, t)$.

The view for the right eye $I_V^R(\varphi + \delta, \theta, t)$ is synthesized similarly by offsetting the camera position along the φ-direction by a certain amount δ. A common rule for stereoscopic imagery is that the maximal angle of divergence between left- and right-eye view should not be excessive. Otherwise, the eyes are forced to diverge to bring distant objects in alignment which causes discomfort. By nature of construction, the direct stereoscopic approach renders converging stereo pairs, and angles of divergence between 0.5 and 1.5 degrees give the most pleasing stereoscopic results.

4.4.3 Depth-Image Based Rendering

An alternative approach is to apply depth-image-based rendering (DIBR) as an intermediate step. Instead of rendering a left- and right-eye view as described in Section 4.4.2, only a center camera view is rendered. In addition, a per-image depth map is obtained. As described in Section 4.4.1, the spatial relationship between corresponding pixels is encoded in the correspondence maps. By assuming linear object motion, it can be determined where an object is located in two different views at the same point in time.

For every pixel position $\mathbf{x}_i$ in A at time t_A, its position $\mathbf{x}_j$ is determined in a neighboring view B (recorded at time t_B). If the two cameras did not capture the images synchronously, the actual position $\mathbf{x}_j''$ of the object is determined by assuming

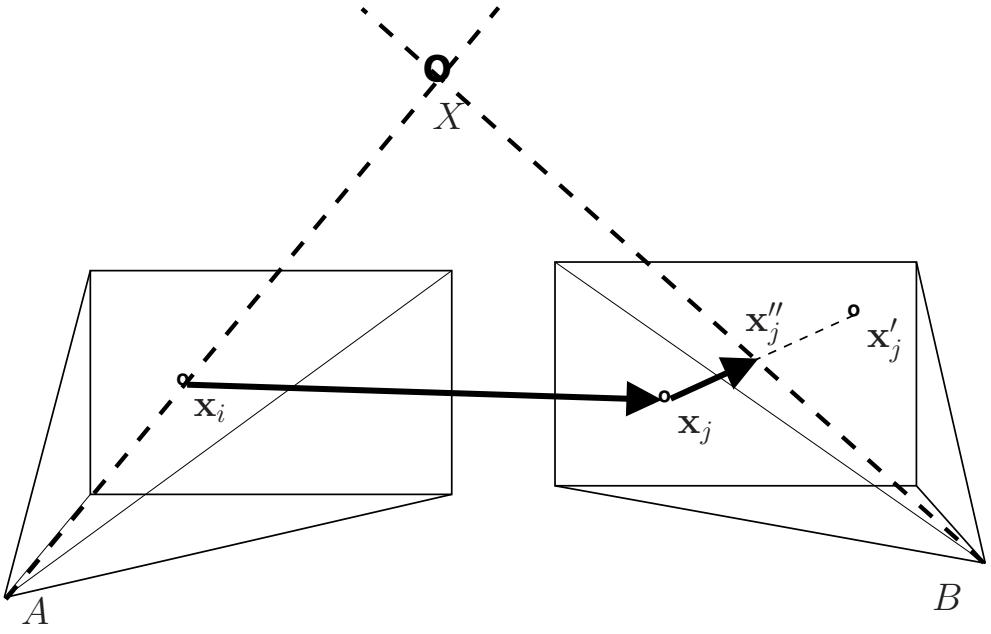


Figure 4.7: Per-pixel depth estimation. When computing the depth of point $\mathbf{x}_i$ in image A, the corresponding position $\mathbf{x}_j$ in image B is determined. If $\mathbf{x}_i$ was observed at time t_A, $\mathbf{x}_j$ was observed at t_B and $t_A \neq t_B$, the temporal correspondences have to be exploited: If the corresponding point $\mathbf{x}'_j$ at time t_{B+1} is known and $t_B < t_A < t_{B+1}$, the temporally aligned corresponding point $\mathbf{x}''_j$ can be estimated. The point X that reprojects into both $\mathbf{x}_i$ and $\mathbf{x}''_j$ is least-squares determined. The depth at $\mathbf{x}_i$ is set to the actual depth of X in A.

linear motion, Fig. 4.7. The three-dimensional location X of this point in Euclidean space is obtained using triangulation in a least-squares sense, i.e., the 3D location is the point with the closest squared distance to the original view rays. This point is reprojected into the original view and the depth value is stored. The depth maps can be used during the following image interpolation phase to resolve occlusion ambiguities.

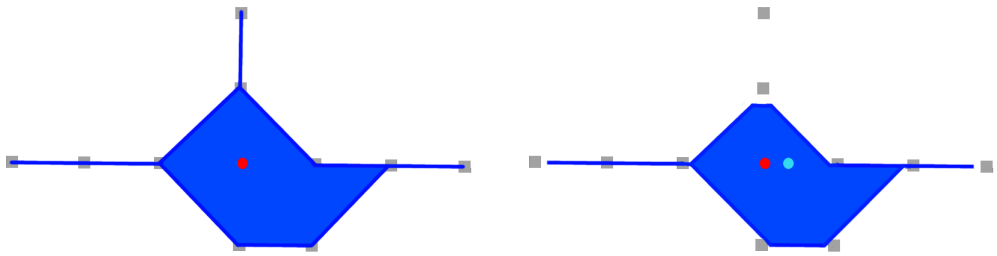
It is also possible to render depth maps from novel views. Instead of interpolating RGB information for every pixel, the Euclidean 3D position of each pixel is interpolated. The location of each pixel is computed on-the-fly in the vertex shader using the depth map and the camera matrix. The final depth value is obtained by reprojecting the 3D position into the coordinate space of the virtual camera.

The color images and depth maps are used to render both a left- and a right-eye view. This is done by using the naïve background extrapolation technique [153].

4.4.4 Comparison

Since both approaches seem appealing at first sight, I would like to compare the individual benefits.

The most striking advantage of the direct approach (Section 4.4.2) is that it does not require any additional building blocks in the rendering pipeline. Basically, the scene is rendered twice with slightly modified view parameters. Since there is no second rendering pass (as there is in the depth-image-based method), no additional resampling or occlusion handling issues arise.



(a) (b)

Figure 4.8: Navigation space boundaries (only spatial dimensions), original camera positions are shown in grey: (a) when a single image (red) is rendered, the full volume (blue) can be accessed. Note that for the horizontal and vertical arcs (left, top, right) only purely horizontal or vertical camera movement is feasible. Otherwise, error-prone long-range image correspondences would have to be used. (b) When an actual camera pair (red/cyan) is rendered, the navigation space volume is effectively eroded, since there has to be a minimum horizontal distance between the both views. This effect prohibits stereoscopic view interpolation if cameras are placed along an vertical arc (b, top).

The most valuable advantage of the depth-based approached (Section 4.4.3) is that is makes no restrictions on camera placement. When using the direct approach (Section 4.4.2), both virtual views must fit into the navigation space boundaries, Fig. 4.8 (b). In contrast, the DIBR method allows stereoscopic rendering even if the cameras are aligned in a vertical arc., Fig. 4.8 (a). In addition, the entire original navigation space can be used: While the DIBR method allows placing the center camera to the far left and far right of the navigation space, the direct method is restricted by the placement of the camera pair, Fig. 4.8. Another benefit becomes obvious when the linear ap-

proximation of object movement is considered. Non-linear object motion may lead to unwanted vertical disparity when the direct method is applied. Although non-linear motion might lead to errors in the depth estimation, the DIBR rendering scheme prevents vertical disparity at the rendering stage. The additional rendering step (to turn color images and depth map of the virtual view into a left- and a right-eye view) might seem as a disadvantage as first. This notion changes if the material is in some way manipulated between these two rendering passes. For example, if any kind of image/video editing operation is applied to the center image, e.g., correction of rendering artifacts, it gets automatically propagated to the left/right eye view. In the case of the direct method, any post-production operations would have to be applied individually (and thus incoherently) to both views.

4.5 Pre-processing and Authoring

Before showing actual results, I would like to explain how real-world data can be processed and which possibilities exist to edit and author the movement of the virtual camera.

4.5.1 Acquisition and Pre-processing

To acquire multi-video data, up to 16 HDV Canon XHA1 camcorders (1440×1080 CCD sensor, 25 fps) are used. The captured sequences are internally MPEG-compressed, stored on DV tape, and later transferred to a standard PC. This setup is very flexible, easy to assemble, runs completely on batteries and is suitable for indoor and outdoor use. Camera arrangement is very flexible and can be adapted to capture arbitrary scenes. Static setups using tripods as well as setups with dynamic hand-held cameras are possible. Adjacent cameras should have sufficient view overlap to facilitate correspondence estimation: Based on the experiences from the *Virtual Video Camera* project, the angle between neighboring cameras should not exceed roughly 10 degrees, in vertical or horizontal direction. Of course, the total range of possible virtual views is determined by camera configuration. Fig. 4.9 shows some typical camera configurations.

Figure 4.9: Camera setups: the proposed approach works with off-the-shelf consumer-grade camcorders, mounted either on tripods or handheld.

Dense Correspondence Field Estimation. In order to interpolate between two images I_i, I_j, bidirectional dense correspondence maps $\mathbf{w}_{ij}$ are needed for each tetrahedral edge in navigation space $\mathcal{N}$, Section 4.2. Theoretically, all image correspondence estimation algorithms that provide visually plausible image correspondences are suitable for this task. For producing early results (Section 4.6), the algorithm proposed by Stich et al. for dense correspondence estimation [178, 179] were employed. For most image pairs, the results are perceptually convincing, and if not, the algorithm accepts manual corrections using a specialized tool, Fig. 4.10. Manual corrections take approximately one minute per image pair. More recent results (Section 4.7) have been produced with the correspondence estimation scheme presented in Chapter 3. In general, the system works just as well with any other correspondence estimation algorithm [210, 150, 154, 202].

4.5.2 Offline Editing: Spacetime Trajectory Editor

So far, I only described how a single in-between view can be synthesized. In order to create complex virtual camera trajectories through space and time, I devise an interactive camera path editor. Its goal is to assign each frame of a desired output video to a point in navigation space. More formally, it should give the user the ability create a mapping

$$\Xi : t_o \mapsto (\varphi, \theta, t_i)$$

Where t_o is a frame of the desired output video and (φ, θ, t_i) is the position in navigation space $\mathcal{N}$. Note that output time t_o and recording time t_i do not necessarily coincide,

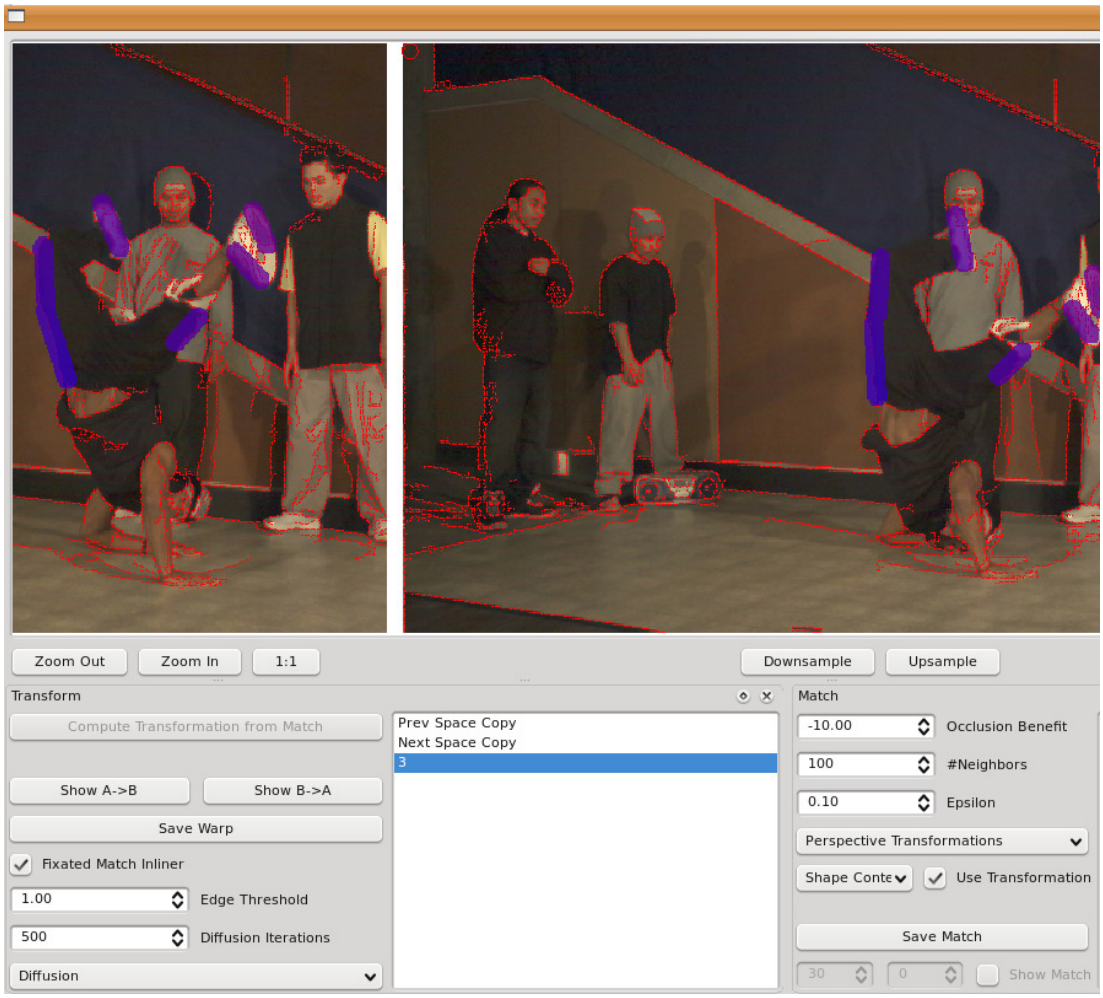


Figure 4.10: Manual correction of correspondence fields: Matching ambiguities in difficult cases (e.g., fast motion in the *Breakdancer* scene) are resolved manually. The user draws corresponding lines onto both images (purple). They serve as prior for the matching process which would otherwise only include automatic correspondence estimation of edge pixels (red). Interaction time per image pair is typically less than 1 minute.

e.g., when time-freeze and slow-motion effects are rendered. A notation similar to the space-time diagrams by Wolf [199] provides the basis for visualization and authoring of the virtual camera trajectory, Fig. 4.11.

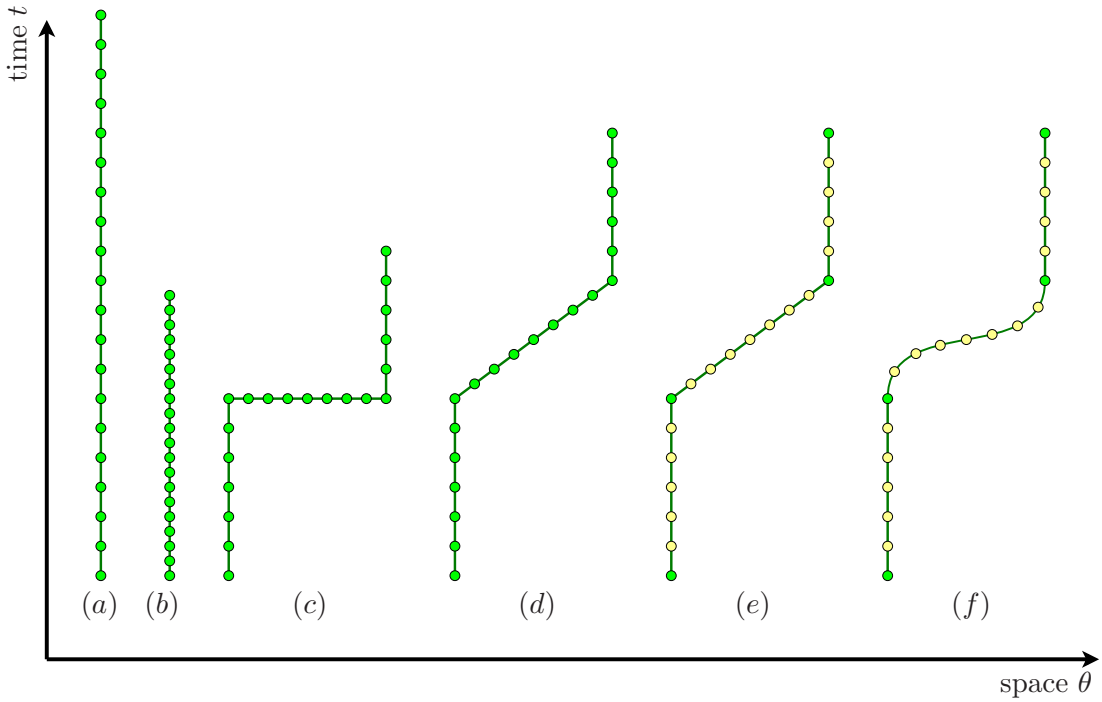
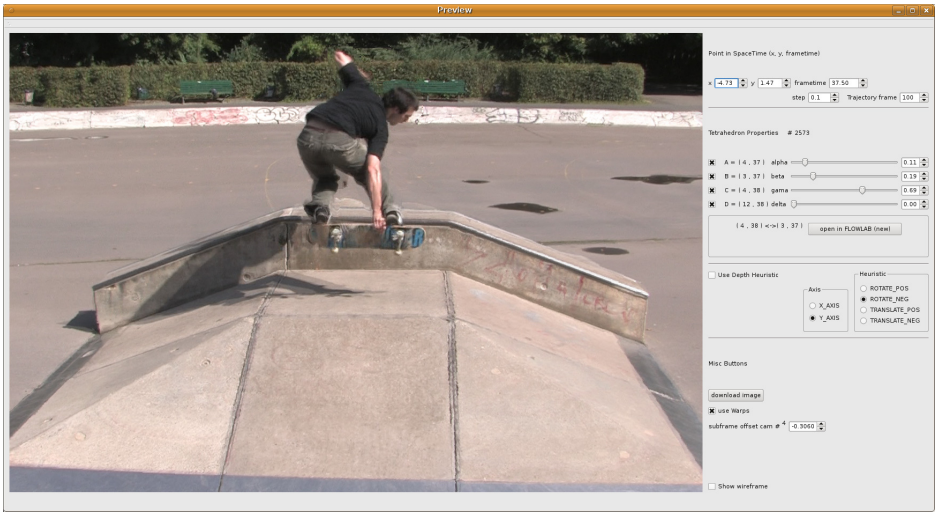


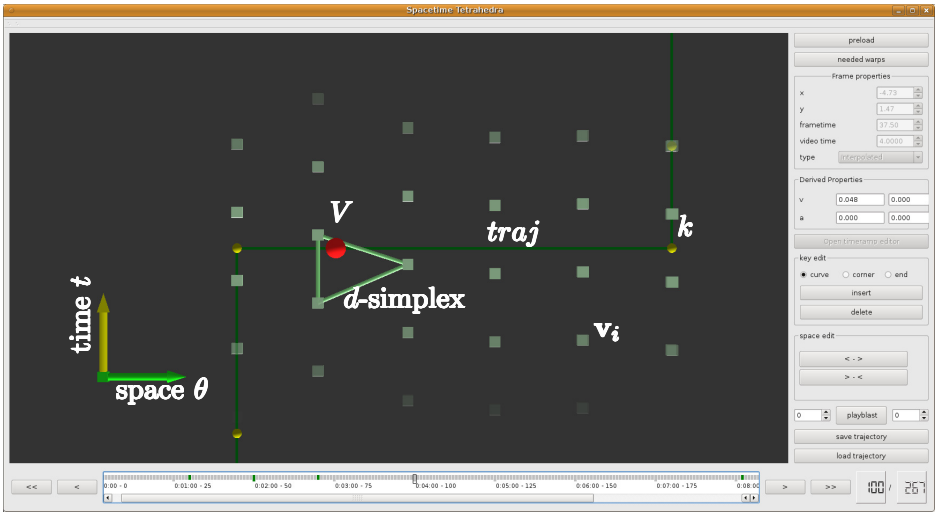
Figure 4.11: Examples for space-time diagrams, based on space-time notation by Wolf [199]. To simplify the diagrams, only one spatial dimension (θ) is displayed. Successive frames are connected by a green line. (a) unaltered, static camera. Camera remains stationary, time sampling matches recording speed of original camera. (b) slow-motion shot, temporal sampling is doubled. (c) time-freeze shot, starting on the seventh frame, camera moves spatially while remaining frozen in time. (d) "bullet-time" shot, similar to time-freeze, but camera movement is combined with slow motion. (e) "bullet-time" shot using key-frame editing. Only green points are defined by the user, other frames are defined by linear interpolation. (f) same as (e), but in-between frames are obtained by Catmull-Rom splines. All configurations depict possible camera trajectories of the same output frame length (20 frames).

The path of the virtual camera in navigation space is shown by the green line. Single output frames are depicted by dots. In order to create free viewpoint videos, the user can specify a space-time position for every output frame. It is also possible to define these points sparsely. The space-time position of in-between frames is determined by linear interpolation or Catmull-Rom splines [30]. This offers a very flexible way to create various space-time effects, such as slow-motion, arbitrary camera pans, time-freeze or

"bullet-time" shots. A screenshot of the editing interface is shown in Fig. 4.12.



(a)



(b)

Figure 4.12: Screenshot of the space-time trajectory editor, seen in two separate windows (a,b). (a) Preview rendering of output frame. On the right-hand side, camera and frame numbers of warped images along with blending weights are displayed. (b) Visualization of navigation space. Notation is inspired by the space-time diagrams of Wolf [199]. In this scene scene, interpolation takes place in two dimensions: horizontal rotation around the scene center (θ) and time (t). Recorded frames are depicted by green squares ($\mathbf{v}_i$). The virtual camera (V) is synthesized by warping and blending the original frames of the enclosing triangle (d-simplex). Camera trajectory ($traj$, green line) is defined by keyframes (k). Output timeline is shown on the bottom of the window, keyframe controls are placed on the right-hand-side.

The editor shows a visualization of the navigation space and a free viewpoint preview rendering. In order to edit the current position in navigation space, the virtual camera can be dragged along the different dimensions in navigation space. Alternatively, a drag-and-drop movement inside the preview rendering moves the virtual camera. Controls exist to insert new keyframes, to change the type of interpolation (linear vs. spline-based interpolation) and to alter the time distance between keyframes. To realize an interactive workflow, correspondence maps are computed on-the-fly using the fast GPU-based optical flow by Werlberger et al. [197]. For the final rendering, the correspondence estimation introduced in Chapter 3 is used. To accelerate the overall process, correspondence estimation can be distributed over a multi-core computing farm.

Wolf [199] also suggests to render frames that integrate over a longer period of time and space. I only consider virtual frames that depict a single time instant and that do not introduce motion or space blur. These and other effects have been investigated by Linz et al. in the context of the *Virtual Video Camera* project [228, 98] and are beyond the scope of this thesis. Various examples of space-time trajectories and associated output frames are shown in Fig. 4.13.

4.5.3 Real-time Editing: Space-time Navigator

Another application is the interactive space-time navigator, Fig. 4.14. While the space-time editor (Section 4.5.2) allows creating sophisticated trajectories through space and time, the space-time navigator is designed to allow for a real-time exploration of the free viewpoint capabilities. To enable real-time rendering, the complete data processing has to be done offline. For example, for a 20-frame slice of the firebreather scene, roughly 3000 image correspondence maps have to be precomputed in advance. The player has been designed with classical video/media players as a basic interaction metaphor, offering a much simpler and more intuitive access to multi-view playback. The user can move camera direction (θ, ϕ) by clicking and dragging the mouse cursor within the rendering window and observe the scene from arbitrary viewpoints. Playback speed can be adjusted by a slider bar, Fig. 4.14 (bottom right). Unlike traditional media players, slow motion playback can make use of temporal interpolation. These capabilities of real-time playback have been further investigated by Meyer et al. [242, 241].

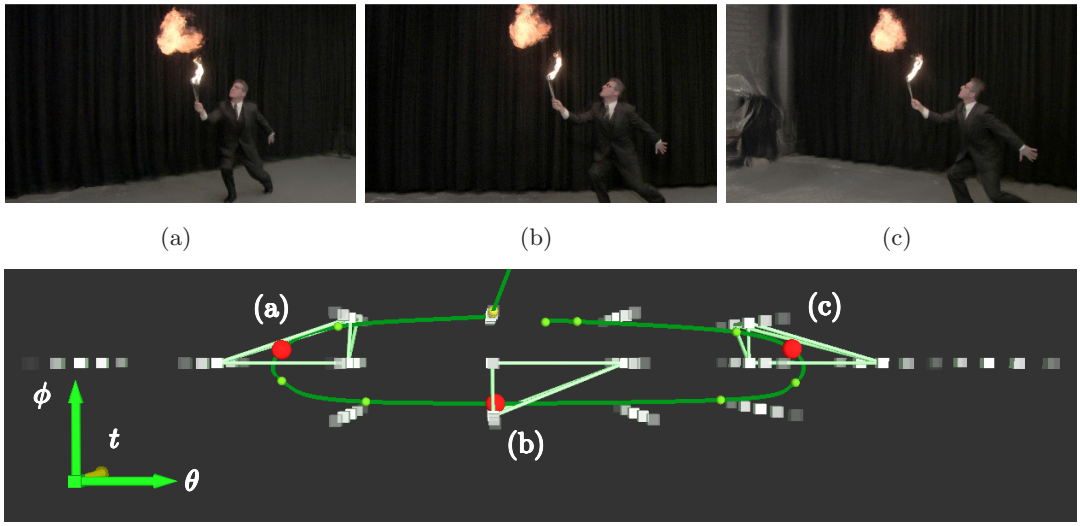


Figure 4.13: (a,b,c) Typical free viewpoint rendering. (d) The virtual camera (red dots) follows a user-defined path (green ellipse). Each virtual view is enclosed in exactly one space-time tetrahedron. In this example, the camera performs an elliptic "frozen time" movement around the scene center.

4.6 Results and Evaluation

I evaluate the system on a variety of real-world scenes, each one posing different challenges to the processing pipeline, Table 4.1. For each test scene, Fig. 4.15 depicts an interpolated view at the barycenter of one navigation-space tetrahedron. The barycenter represents the point where all four video images are warped and weighed equally, thus representing the case most likely to show rendering artifacts. Because still images are not able to convey the full impression of interactive viewpoint navigation, I refer to the online video resources and the interactive Space-time Navigator [34]. Besides high-quality image interpolation in space and time, the design of navigation space allows intuitive exploration of the recorded space-time. As such, the system is a valuable tool for intuitive design of visual effects such as time-freeze and slow motion in post-processing.

Besides scene-specific challenges, the system must also cope with lens distortion, camera noise, and compression artifacts inherent to consumer-grade camcorders. The *Juggler* sequence was recorded using hand-held camcorders with the cameramen moving about 0.5 meters to the left during acquisition. Nevertheless, a static virtual view

Figure 4.14: Interactive Spacetime Navigator: while watching the scene, the user can control playback speed and view perspective by clicking and dragging within the video window. Rendering frame rates exceed 25 fps on a NVIDIA GeForce 8800 GTX graphics card.

can be synthesized. The small geometrical error introduced by the non-stationary approximation of the camera arc manifests itself in a slight drift of the virtual camera, yet it is hardly noticeable. In order to remedy this undesired effect, further stabilization techniques can be integrated [116]. The *Breakdancer* sequence is provided for visual comparison to the approach of Zitnick et al. [211]. As their data set is closely sampled in the spatial domain (the angle between adjacent cameras is approximately 3 degrees), no user interaction is required for inter-camera correspondence field estimation. The video is not perfectly matched to the camera movement of the original video, since the original space-time trajectory is unknown. I also decided not to imitate the original cross-blending effect, since it distracts from potential rendering artifacts. While obtaining comparable visual quality, interpolation is also possible along the temporal dimension. Unfortunately, the *Breakdancer* data set was captured at only 15 fps and exhibits very fast rotational movement, which violates the assumption of linear motion. Still,

spatiotemporal image interpolation can be achieved. However, most of the inter-frame foreground correspondences had to be edited by hand, since the automatic matching cannot cope with the fast motion of the breakdancer.

Limitations. Similar to disparity-based free viewpoint navigation systems [211], the virtual viewpoint is spatially restricted: The user can viewpoint-navigate on the hull spanned by all camera recording positions, looking at the scene from different directions, but he cannot, for example, move into the scene or fly through the scene.

Output rendering quality obviously depends on the visual plausibility of the correspondence fields. While the pair-wise correspondence estimation algorithm by Stich et al. [178, 179] yields convincing and robust results overall, human interaction is explicitly allowed to correct for remaining spurious correspondences. While for some scenarios other specifically tailored matching algorithms might yield better results, I incorporated a matching algorithm that provides robust results for most scenes. In some cases, manual correction is not feasible. For example, the *Beer* scene features single streaks of foam that cannot be matched at all. In the case of fluids, i.e., in the *Water* sequence, it can be observed that automatic feature matching works quite well for freeze-and-rotate shots. Bearing in mind that the short *Firebreather* sequence contains more than 3000 pairwise correspondence fields, the vast majority of correspondences in the presented results is computed without user interaction. The same is true for the rendered videos, e.g., the trajectory of the virtual camera in the *Dancer* scene requires several hundred correspondence fields. Editing more than one or two dozen correspondence fields per scene is not feasible. Small uncorrected inaccuracies in automatically computed fields become manifest in cross-fading/ghosting artifacts visible in some scenes.

Occlusion is handled by heuristics as proposed by Stich et al. [178, 179], disocclusion is handled by mesh cutting based on the connectedness of the correspondence fields [124]. In cases where these occlusion heuristic fail or correspondence fields are faulty, visible rendering artifacts occur at occlusion borders, e.g., the small ball in the *Juggler* scene and the silhouette of the skateboarder are not preserved faithfully. More sophisticated occlusion handling techniques, such as presented by [71], [80] or [123] might improve rendering quality in these cases.

The interpolation model currently employed is applicable to scenes that feature only one motion direction per pixel: Currently, overlaying semi-transparent objects

that move in different directions are not accounted for. For high-quality rendering results, it can be observed that the angle between adjacent cameras should not exceed 10 degrees, independent of scene content, Table 4.1. For greater distances, missing scene information appears to become too large to still achieve convincing interpolation results. The same is true for too fast scene motion. As a rule of thumb, scene correspondences should not be farther apart than approximately 10% of linear image size.

Scene	**Camera Setup**	**Challenges**
Dancer	top: 4 cameras middle: 8 cameras bottom: 4 cameras	wide-angle lens distortion, fast motion
Water	5 cameras in half-circular arc	refraction, reflection, specular highlights
Beer	top: 5 cameras middle: 5 cameras bottom: 5 cameras	high speed motion, changing topology
Fire-breather	top: 4 cameras middle: 8 cameras bottom: 4 cameras	high dynamic contrast, over-exposure, volumetric effects
Skate-boarder	6 cameras in half-circular arc	outdoor capture, varying lighting conditions
Break-dancer	8 cameras in half-circular arc	very fast motion, low temporal sampling, noisy acquisition
Juggler	5 cameras in half-circular arc	outdoor capture, hand-held and moving cameras

Table 4.1: Camera recording arrangements. Cameras are spaced approximately 10 degrees apart in horizontal as well as vertical direction (3 degrees in the Breakdancer scene). I have deliberately chosen non-trivial test scenes to assess performance for a range of different scenarios.

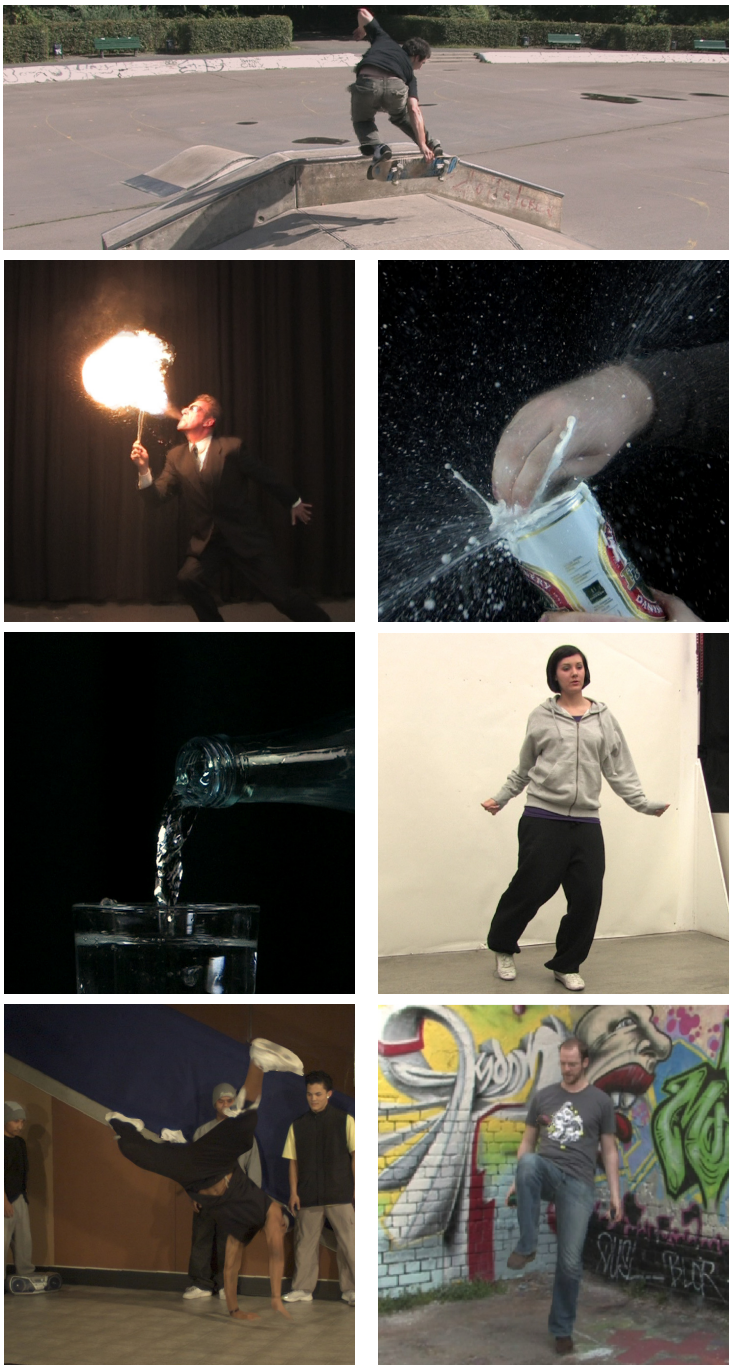


Figure 4.15: *Virtual Video Camera* views: for each test scene, the virtual viewpoint at the barycenter of the enclosing navigation-space tetrahedron is rendered, representing the 'worst case interpolation'. As test scenes, I have deliberately chosen complex scenes that pose a variety of challenges: outdoor capture (*Skateboarder*), volumetric effects and high dynamic contrast (*Firebreather*), fast, non-rigid motion (*Beer*), reflection and refraction (*Water*), wide-angle lens distortion (*Dancer*), very fast scene motion (*Breakdancer*), and hand-held acquisition (*Juggler*).

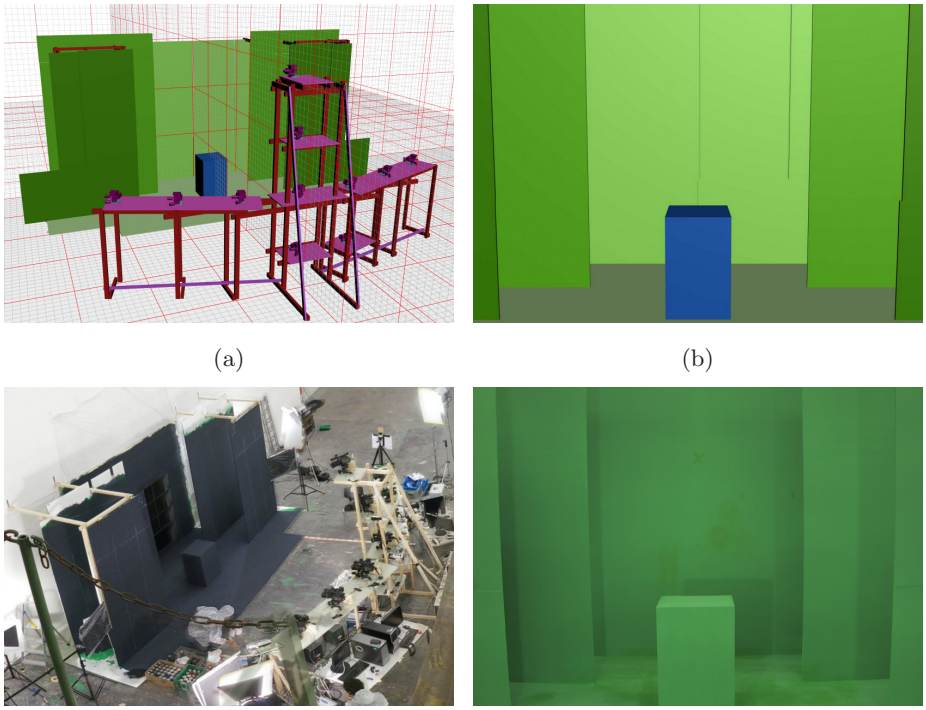
4.7 Video Production Pipeline Integration: “Who Cares”

So far, I concentrated on giving short demonstrations to showcase the technical possibilities of the proposed free viewpoint rendering approach. In this section, I give a detailed account of how free viewpoint rendering was used for the production of the “Who Cares” music video. The music video was created as a collaborative effort of our research group at TU Braunschweig and the *Institut für Medienforschung* (Institut for New Media) at the *HBK Braunschweig* (Braunschweig University of Arts). On our side, Felix Klose and Kai Ruhl contributed to the project. While my task was the integration of the *Virtual Video Camera*, my colleagues worked on the capturing software, the processing of the depth data [246, 248] and tools for manual correction of the correspondence maps [224, 247].

Although convincing results had been created prior to this project, it was the first time we embraced the challenge of using our technology in the context of an actual production. The main goal of the project is not just to showcase the free viewpoint technology, but to prove that it is a helpful tool that supports the creative process of movie production. Several aspects of our technology are very appealing: First of all, the possibility to create arbitrary camera motions in post-production allows iterative modifications of the camera path until it optimally supports the music playback. Different speeds and accelerations can be used for moving the camera, which is hard or impossible to recreate with a practical setup, e.g., dolly tracks or a crane. Second, the possibility to interpolate in the temporal domain enables sophisticated time-freeze and slow-motion effects.

It also proved to be very helpful to use previsualization techniques for planning the set and the camera setup, Fig. 4.16. By constructing the set prior to the actual production, the possible movement of the virtual camera can be explored freely. This helps to determine the size of the set and the camera rig as well as the amount and spacing of cameras.

Project Idea. In the “Who Cares” music video, HD input material is processed that features two timelines (live foreground action and background timelapse) and is captured with non-synchronized camcorders. It is the first project that makes use of this kind of input material and allows horizontal, vertical and temporal image interpolation.



(a) (b)

(c) (d)

Figure 4.16: Previsualization and actual setup. (a) Previsualization of the film set and the camera rig was done to develop the look of the video and for planning the actual construction. (b) Possible free viewpoint camera angles could be explored prior to shooting of the video. (c) The practical set was built within two days including setup of cameras and recording hardware (4 Linux PCs and external HDs). (d) Previsualization proved to be a versatile tool to predict the composition of the recorded material.

The basic concept is that two DJs appear on a stage and perform a live act with their audio controller, Fig. 4.17 (upper right). During the course of the music video, the background is painted over with various graffiti motifs (e.g., giant stereo speakers, equalizer bars or a night skyline), Fig. 4.17 (upper left). Although it would be possible to create and animate these graffiti motifs with traditional CGI tools, it is an artistic decision to use an actual graffiti timelapse to maintain a credible "low-fi" look. By collaborating with the film makers who are familiar with traditional 2D content post-production, the possibility to seamlessly integrate the free viewpoint system into an existing pipeline could be explored.

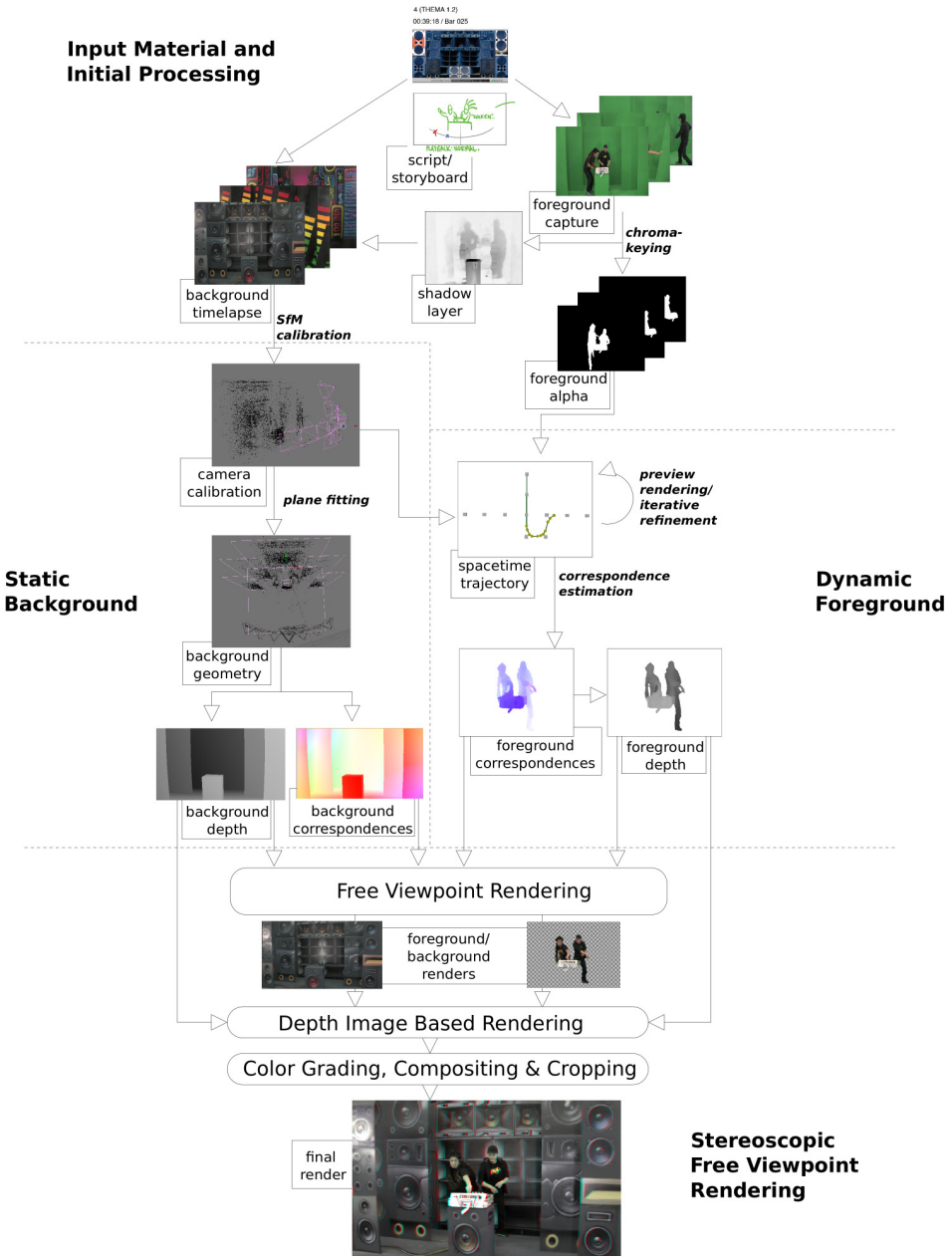


Figure 4.17: An overview of the production pipeline. After an initial processing stage (top), foreground and background footage is processed independently. While background depth and correspondence maps are derived from a static geometric model (left), foreground data is computed using state-of-the-art correspondence estimation algorithms (right). The processed footage is passed to a free viewpoint renderer before the actual left- and right-eye view are rendered using depth-image-based rendering (bottom).

Input Material and Initial Processing. The first step in the project is the storyboard that defines a rough mapping between the different graffiti motifs in the background and a basic choreography for the actors in the foreground. In addition, the approximate position and movement of the virtual camera is sketched, Fig. 4.18. The

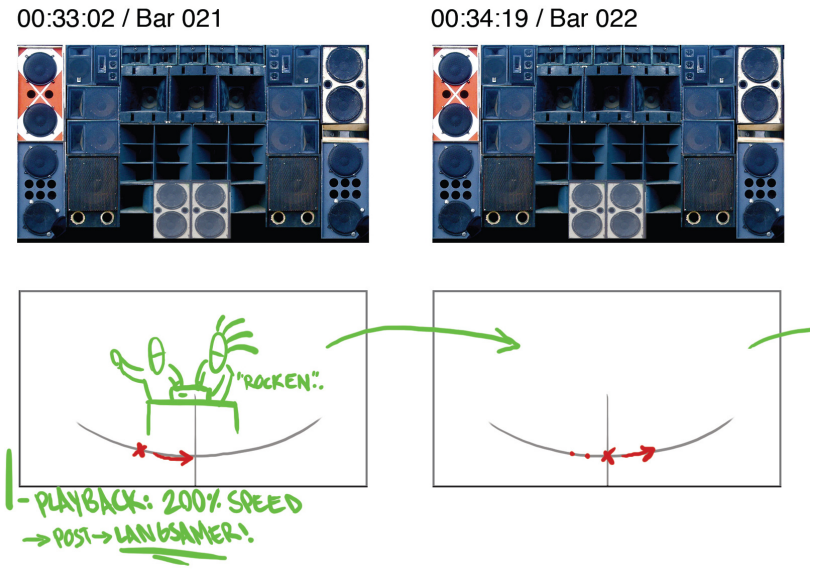


Figure 4.18: Storyboard of "WhoCares". During pre-production, a new notation for the space-time effects was devised for this particular project. On top, the current time within the audio track is defined. Below, the current background graffiti motif is displayed. Dancing instruction of the actors is visualized (green). The movement of the virtual camera is indicated on the bottom cross (red).

basic movement of the camera is either left-to-right or up-and-down. In some parts of the video, transitions between these two predominant moves are inserted. On some occasions, more complex two-dimensional fly-throughs of the virtual cameras are needed (e.g., the timefreeze scene).

For the recordings, a custom-built capturing software is used that runs on four Linux PCs and controls 11 Canon XHA-1 HD camcorders at 1440 × 1080 px, 25 fps. Although the capture PCs are interconnected via ethernet, the dv1394 camera interface does not allow an accurate synchronization of the cameras. In addition, 11 Canon 5D Mark 1 DSLRS and 2 Microsoft Kinect RGB-D sensors were attached to the camera rig. The footage of these sensors has not been used for the creation of the music video, but served as a basis for further multi-view video research [246, 248].

The foreground action is shot in front of an all-green background. Chroma-keying is used to extract an RGBA representation as well as a “shadow layer” of the foreground using Adobe After Effects and Keylight. The shadow information is obtained by investigating brightness differences in the background areas between the actual shots and a clean-plate version of the background.

After completion of the foreground capture, the graffiti timelapse was recorded over a period of five days. The capturing software was set to a timelapse mode and unwanted frames were removed manually. Using conventional 2D animation techniques, some animation effects, e.g., pumping equalizer bars or vibrating boombox speakers are inserted into the footage.

Although manual white-balance is applied to all cameras, some color balance differences can be observed in the material. These are corrected in Adobe After Effects. For this particular project, a production pipeline that processes fore- and background independently is devised, Fig. 4.17.

Static Background. First, an off-the-shelf structure-from-motion system [173] is used to obtain the intrinsic and extrinsic parameters of the cameras. To improve the calibration, the SIFT feature matching scheme is modified so that features of different graffiti motifs are used for the same calibration run. This resulted in a significantly larger set of matched features between images. As described in Section 4.2, the Euclidean 3D positions of the cameras are used to parametrize their horizontal and vertical position in navigation space $\mathcal{N}$. The resulting camera parameters and the sparse reconstruction of the scene geometry are imported into Blender [17]. The scene geometry is fitted manually to the reconstructed point cloud. Although methods exists to perform this task automatically, e.g. [156], the simplicity of the scene geometry makes the manual approach feasible.

The camera calibration together with the reconstructed scene geometry allows to compute depth maps for each view and also correspondence maps between the different camera positions. Using OpenGL and GLSL shaders, both correspondence and depth maps are rendered.

Dynamic Foreground. The calibration data obtained in Section 4.7 together with the foreground footage allows to turn the roughly defined camera movement from the

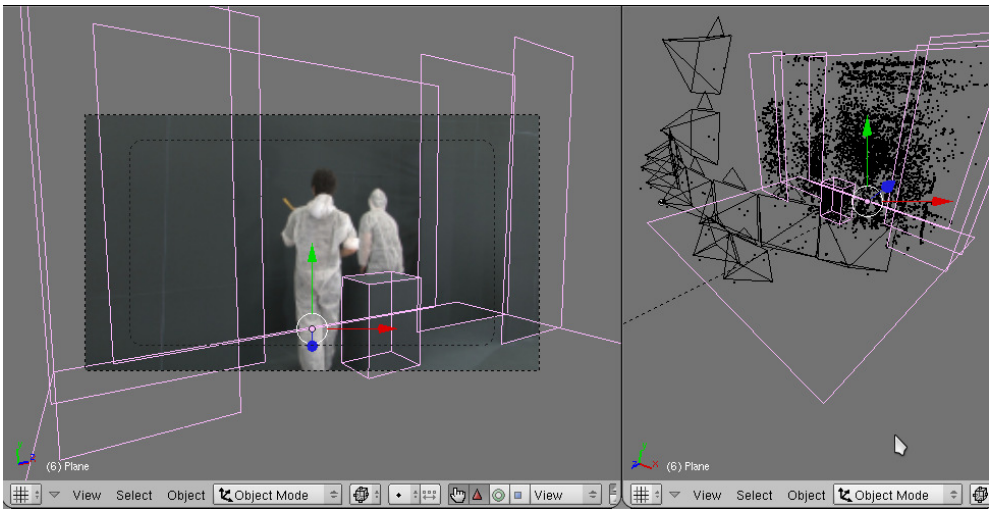


Figure 4.19: Fitting a geometric model to the static background. The camera orientations and a sparse scene geometry are imported into Blender (right). Single quads are fitted manually to the point cloud. Using the actual footage as reference (left), a pixel-exact geometric proxy consisting of 10 faces is created.

script into a valid space-time trajectory. Using the graphic trajectory editor (see Section 4.5.2), key points in navigation space $\mathcal{N}$ are mapped to every dedicated frame in the final video. For in-between frames, linear interpolation or Catmull-Rom splines are used. Since the free viewpoint framework allows to change the trajectory freely in post-production, many variations of the camera trajectory can be explored. In order to get live feedback, low-res, foreground-only preview renderings are used. Since no high-quality correspondence maps of the foreground are available at this stage of the production pipeline, the real-time optical flow by Werlberger et al. [197] is employed.

After choosing the final trajectory, high-quality correspondence maps are computed using the approach described in Chapter 3. We used a 30-core Linux cluster to obtain the final correspondence maps. In order to increase computation speed, correspondence values are only computed on pixels with non-zero alpha values. Spurious mismatches between images are corrected with the tool presented by Klose et al. [224].

Afterwards, a depth map is computed for every image needed for rendering, cf. Section 4.4.3. If more than one neighboring view is available, depth values are averaged. Since the resulting disparity does not exceed a few dozen pixels, byte-precision depth maps are sufficiently accurate.

Stereoscopic Rendering and Compositing. Incorporating the original foreground and background images, the correspondence and depth maps as well as the space-time trajectory, both color and depth maps are rendered for foreground and background. We used GIMP to manually correct rendering artifacts, e.g. streaking and ghosting, in some images. Finally, depth-image-based rendering is used to obtain a left- and right-eye view for each frame.

As a last step, foreground, shadow layer, and background are composited and a final color grading is applied. Since rendering artifacts typically occur at the image borders, these regions are cropped (10% of image width and height are discarded). The final video is downsampled to 1280 × 720 px to account for the slight blurring introduced by the forward-warping-and-blending scheme of the renderer.

Using the stereoscopic free viewpoint video framework, a 3-minute music video is rendered at a resolution 1280 × 720 px, 25p, Fig. 4.20 shows a selection of renderings. Except for the intro (∼40 seconds) and outro (∼30 seconds), free viewpoint rendering is used for the whole length of the video. A total of more than 3000 bidirectional correspondence maps had to be computed, each one took about 10 minutes on a single core. The final video is available online at `http://graphics.tu-bs.de/projects/whocares`.

Lessons Learned. The “WhoCares” music video proves that the *Virtual Video Camera* can be integrated into an actual (music) video production. Free viewpoint rendering can deliver more than just short technical demonstrations, but can can play a vital role by supporting the artists in their creative process. Most importantly, film makers are not forced to create their imagery inside the computer, but can use free viewpoint rendering as a powerful tool to fully control the (virtual) camera movement in real-world scenes. In contrast to previous projects, this has raised awareness outside the research community: “WhoCares” has been presented at numerous professional events (e.g., FMX 2012 [83], Hands on HD 2011 [103]) and has been covered by specialized press (FKT [239], fxguidetv [132]).

Since the quantitative and qualitative requirements were formulated by artists and not by researchers, the full artistic potential could be leveraged. Also, remaining practical shortcomings can be identified:

- The need for manual correspondence field correction [224] means that additional effort is required to obtain visually convincing results.
- Novel views can only be placed inside the navigation space spanned by the 11 HD camcorders, Fig. 4.8. As discussed earlier (Section 4.4), this complicates stereoscopic rendering and view extrapolation.
- Additional depth sensors (e.g., the two Kinect RGB-D sensors used on the "Who-Cares" set) cannot be incorporated into the reconstruction or rendering process.

To a large extent, the formulation of the hybrid rendering presented in the next chapter is motivated by these limitations.

Figure 4.20: Results: Exemplar stills from the “Who Cares” music video. Best viewed with red/cyan (left/right) anaglyph glasses.

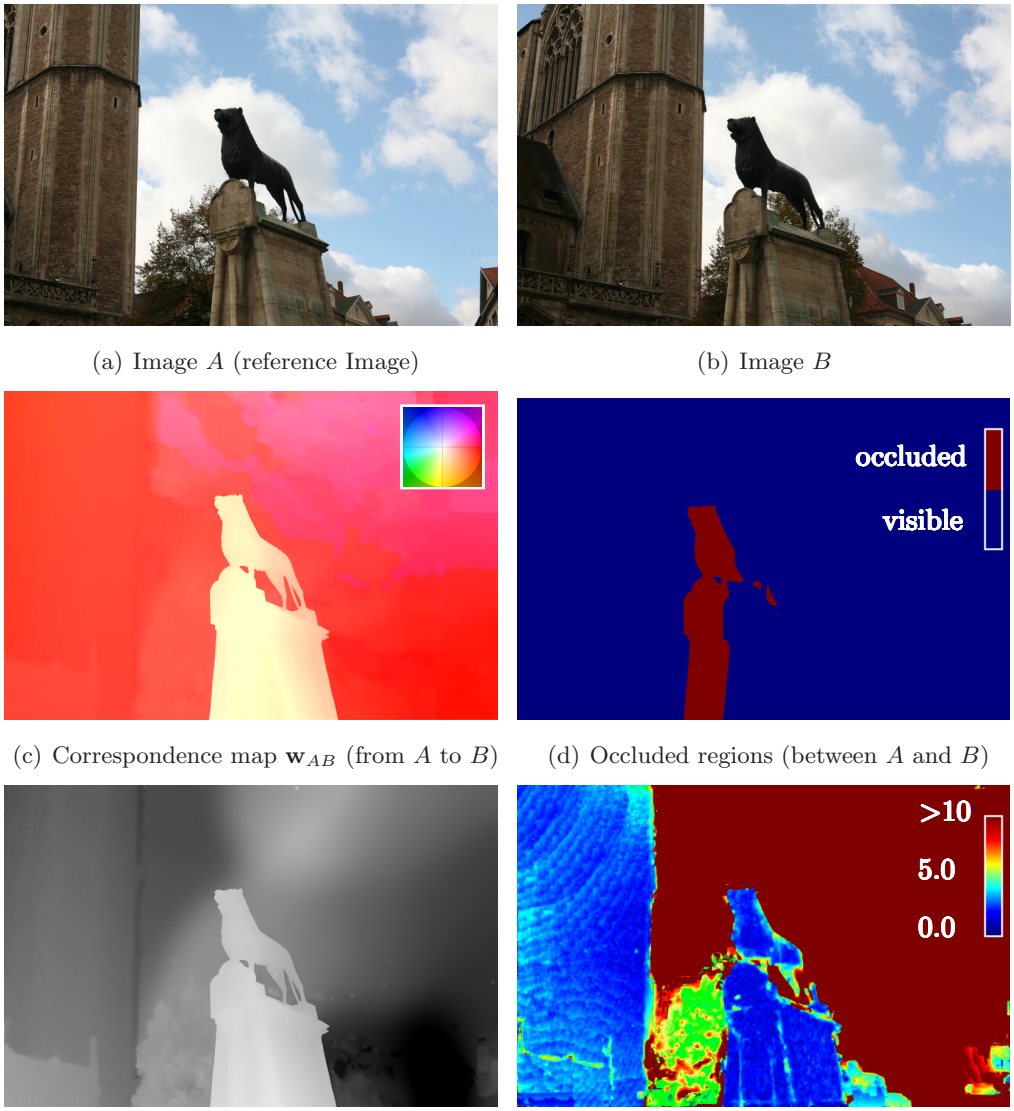
5

Correspondence and Depth-Image Based Rendering

Let us briefly revisit the capabilities of correspondence-based renderings described in the previous chapter. Using dense correspondence maps, plausible in-between views can be rendered even for non-synchronized recordings. The integration of this approach into a full-length music video production (Section 4.7) demonstrates its general applicability.

Still, some limitations have to be overcome: The correspondence-based rendering equation (Section 4.3) is not capable to extrapolate the viewpoint. The view extrapolation step used for stereoscopic rendering in Section 4.4 is done in post-processing and only allows small deviations from the original navigation space boundaries. It is not possible to composite correspondence-based free viewpoint renderings with other images that have been captured from a view outside the camera manifold. The ability to alter or edit scene content is very limited: E.g., rendering of shadows onto arbitrary geometry or relighting of the scene is not possible. Since automatic correspondence estimation is still error-prone, it fails in some cases and makes manual editing of the data necessary.

As discussed in Chapter 2, geometry-based or depth-based renderers are able overcome these limitations to some degree. So far I did not exploit this property, since the epipolar constraint does not hold true when working with "casually" captured data: Errors in camera calibration or moving scene content may make depth reconstruction impossible. However, depending on scene content and capture modalities, scene geometry may be reconstructible for some parts of the image.



(a) Image A (reference Image)

(b) Image B

(c) Correspondence map $\mathbf{w}_{AB}$ (from A to B)

(d) Occluded regions (between A and B)

(e) Depth map of A

(f) Reprojection error [px]

Figure 5.1: Problems with "casually" captured data. (a,b) Two images of a multi-view recording. (c) When computing pairwise correspondence maps between images A and B, (d) occluded parts (background occluded by central statue) cause problems for correspondence-based rendering (CBR) during correspondence estimation as well as rendering. (e) When trying to reconstruct depth from multi-view recordings via triangulation of corresponding pixels in other images, (f) the reprojection error is high for moving objects (tree and clouds), making depth-image-based rendering (DIBR) hard or impossible. I strive for a rendering and reconstruction framework that combines the strength of both approaches.

Let me illustrate this with a practical example: We observe two images of an outdoor scene containing a central lion statue, some buildings, trees and a clouded sky, Fig. 5.1 (a,b). The pictures were captured with a handheld DSLR camera from different angles. Approximately seven seconds passed between the images.

First, we compute pairwise correspondence maps between two adjacent views A, B. Let us assume that we can successfully identify corresponding pixels, Fig. 5.1 (c). For parts of the image, however, pairwise correspondences cannot possibly be estimated because they are not visible or occluded by some foreground object (e.g., the central statue) in the other view. These areas are visualized in Fig. 5.1 (d, red area). When rendering intermediate views between A and B, these regions are problematic since we cannot determine where they should be displayed. Although we can estimate correspondence values for small occluded regions, Section 3.1.5, large occluded regions still pose a problem.

We try reconstructing depth data of a reference image A by computing correspondences $\{\mathbf{w}_{AB}, \mathbf{w}_{AC}, \mathbf{w}_{AD}, \ldots\}$ to the other images $\{B, C, D, \ldots\}$ of the data set and then triangulating the 3D position of the individual pixels, Fig. 5.1 (e). As commonly used in computer vision, we evaluate the reprojection error of the 3D point as a measure of reconstruction fidelity: The central lion statue as well as the building in the background have a low reprojection error, meaning that these parts of the scene can be handled well by depth or geometry-based approaches, Fig. 5.1 (f, blue areas). The tree in the background performs slightly worse as the reprojection error is significantly higher, Fig. 5.1 (f, green area). One can assume that the branches had been waving in the wind and slightly moved between captures, violating the epipolar constraint. Still, plausible depth values can be obtained for most parts of the vegetation. A different behavior can be seen in the clouds of the background. Although a visually plausible correspondence map $\mathbf{w}_{AB}$ can be computed, the calculated depth values must be discarded as invalid due to their high reprojection error, Fig. 5.1 (f, red area). This is not a surprise since the clouds had been moving across the scene for several seconds between the captures: According to the DWD (German Weather Service), near ground wind speeds of up to 49 km/h (strong breeze) were recorded on the day of capture (15 Sept 2011) in this area (Northern Germany) [40].

When producing free viewpoint video of such a scene, we face a dilemma: Depth-image-based rendering (DIBR) would surely allow us to believably re-render the static

content of the scene and maybe even the moving foliage. However, large parts of the sky would have to be discarded, since no valid depth values can be obtained. Complementary, correspondence-based rendering (CBR) is able to render in-between views of all scene parts but suffers from the principal limitations discussed in Section 4.6: Since the two images contain large occluded areas, a plausible image-based rendering result is hard if not possible to achieve for these image parts. Additionally, view extrapolation is not possible out-of-the box, as discussed in Section 4.4.4.

In this chapter, I present a novel approach to free viewpoint video that addresses the shortcomings of CBR and DIBR. The main contribution is the formulation of a hybrid approach between CBR and DIBR. When rendering the scene from novel viewpoints, both dense pixel correspondences between image pairs as well as an underlying, view-dependent geometrical model are used. The novel reconstruction scheme iteratively refines geometric and correspondence information. By combining the strengths of both depth and correspondence estimation, the novel approach enables free viewpoint video rendering for challenging scenes as well as for recordings that may violate typical constraints in multi-view reconstruction. The key idea is to apply image morphing in 3D space: I expect geometric scene information to be approximate, faulty or even missing. Therefore, I compensate for inaccuracies and invalid assumptions made in reconstruction by aligning image regions according to bidirectional correspondence maps. The method is robust against inaccurate camera calibration, asynchronous capture, and imprecise depth reconstruction. Rendering results for different scenes and applications demonstrate the versatility and robustness of the approach.

I contrast the basic properties of CBR and DIBR in Section 5.1. In Section 5.2, the first main contribution is introduced: A hybrid rendering equation is presented that incorporates both CBR and DIBR, Fig. 5.2. This enables to significantly reduce rendering artifacts in scenes that violate common assumptions made in DIBR (e.g., static scene content, epipolar constraint). The second main contribution is the iterative scheme for joint reconstruction of image correspondences and depth, Section 5.3. The key element in reconstruction is a soft geometric constraint that enforces correspondences to be consistent with sparse geometric information. From these correspondence maps, robust, dense depth maps are computed. The geometric constraint is updated accordingly and correspondence maps are re-estimated, Fig. 5.3.

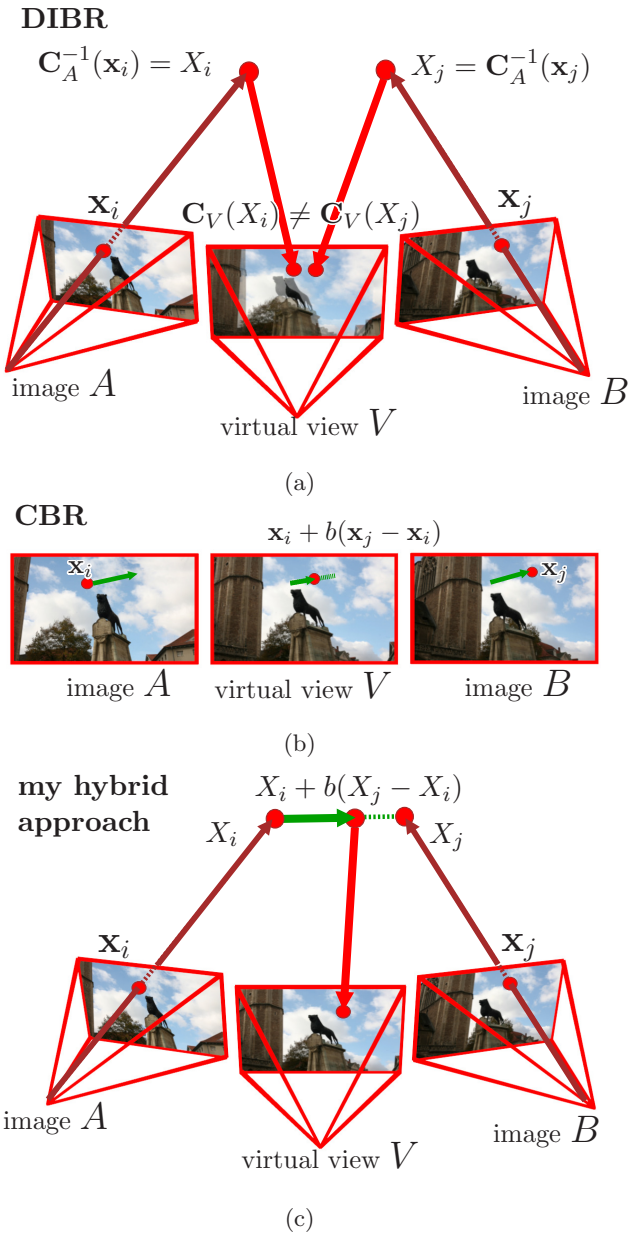


Figure 5.2: Correspondence and depth-image-based rendering (CDIBR). (a) In depth-image-based rendering (DIBR), corresponding pixel positions $\mathbf{x}_i$, $\mathbf{x}_j$ in images A, B are projected to their estimated world space positions X_i, X_j. Because of reconstruction inaccuracies and invalid assumptions, reprojections of X_i, X_j might not align in a virtual view V. (b) In correspondence-based rendering (CBR), pixel correspondences are directly used to create an in-between view V. (c) In my novel approach (CDIBR), world space positions are reconstructed and pixel correspondences are used to align corresponding points X_i, X_j in world space.

In Section 5.4, the technical details of the rendering pipeline are presented. Different data sets and applications are showcased in Section 5.5.

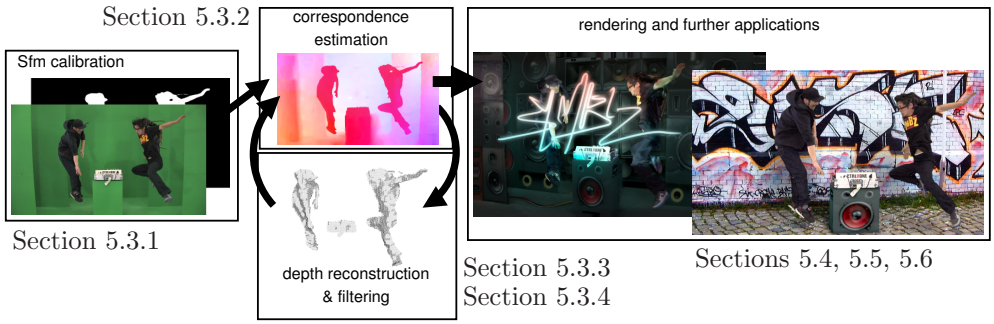


Figure 5.3: Overview of reconstruction and rendering pipeline. After camera calibration and initial (sparse) depth reconstruction (Section 5.3.1), correspondences are estimated for neighboring images (Section 5.3.2). Depth is reconstructed using triangulation (Section 5.3.3). Since depth inaccuracies and holes can be expected, a filtering is applied to the depth maps to obtain robust results (Section 5.3.4). Correspondences are refined using reconstructed depth as a constraint. Both depth maps and correspondences are used for rendering (Section 5.4). I showcase the proposed approach on different data sets (Section 5.5) and applications (Section 5.6), allowing composites of free viewpoint renderings with 3D objects and other camcorder captures.

5.1 Mathematical Foundations

Let us briefly revise the unique strengths and features of both depth-image as well as correspondence-based rendering. In depth-image-based rendering (DIBR) [46], for any given pixel location (u, v) in an image A, not only the color information $I_A(u, v) = (r, g, b)$, but also depth information $D_A(u, v) = d$ is available. The image space location $\mathbf{x}$ in an image A is defined as $\mathbf{x} = (u, v, D_A(u, v), 1)$. Along with intrinsic and extrinsic camera parameters, the original 3D location of the pixel can be reconstructed, as the depth $D_A(u, v)$ is known. In order to synthesize the view of a virtual image V, each pixel of an image A at a given location $\mathbf{x}$ has to be reprojected to the image plane of V.

$$\tilde{I}_A(\mathbf{C}_V(\mathbf{C}_A{}^{-1}(\mathbf{x}))) = I_A(\mathbf{x}) \tag{5.1}$$

$\mathbf{C}_A$ denotes the projection from 3D world space to the image space of image A and $\mathbf{C}_A{}^{-1}$ is the inverse projection, Fig. 5.2 (top). $\mathbf{C}_A$ is defined by

$$\mathbf{C}_A(X) = \mathbf{A}_A\mathbf{P}_n\mathbf{D}_A(X) \tag{5.2}$$

where X is a world space point in homogeneous coordinates, $\mathbf{D}_A$ and $\mathbf{A}_A$ are the extrinsic and intrinsic matrices associated with image A, and $\mathbf{P}_n$ is the normalized perspective projection matrix [201].

Typically, more than one image is used for image synthesis, e.g., the rendered image can be a combination of two reprojected images A, B. In order to render V, its extrinsic and intrinsic camera parameters $\mathbf{D}_V$ and $\mathbf{A}_V$ as well as weighting coefficients a, b with $a + b = 1, \quad 0 \leq a, b \leq 1$ have to be provided. Both input images are reprojected according to (5.1) and blended according to weights a, b:

$$I_V(\mathbf{x}) = a\,\tilde{I}_A(\mathbf{x}) + b\,\tilde{I}_B(\mathbf{x}) \tag{5.3}$$

Another possibility to render in-between views is correspondence-based rendering (also known as image warping or image morphing). In contrast to DIBR, image synthesis is done in image space, Fig. 5.2 (middle). When rendering a virtual view V between images A and B, for every pixel location $\mathbf{x}$ in A, the corresponding location $\mathbf{x}+\mathbf{w}_{AB}(\mathbf{x})$ in B must be known (and vice versa). $\mathbf{w}_{AB}$ is a so-called correspondence map which stores for each given pixel position $\mathbf{x}$ in A the vector pointing to the corresponding location in B. Since V is represented as a weighted combination of A and B, this correspondence information can be used to warp each pixel towards its corresponding counterpart:

$$\tilde{I}_A(\mathbf{x} + b\,\mathbf{w}_{AB}(\mathbf{x})) = I_A(\mathbf{x}) \tag{5.4}$$

It is important to understand the differences between both approaches. DIBR provides the ability to extrapolate viewpoints. Since the placement of the virtual camera is arbitrary, any virtual image V, with a camera orientation reasonably close to one of the original images, can be rendered. The availability of depth information can also be exploited to resolve occlusion ambiguities during rendering. DIBR makes strong assumptions regarding camera calibration, capturing modalities and scene appearance. Often, a precise calibration of all cameras is needed. In addition, scene content has either to remain static during acquisition, or multiple cameras have to be triggered synchronously. Another assumption is that visually identical image regions depict the same scene object in world space. This assumption can be violated, a prominent example are the silhouettes of objects [29]. Although they can be matched between images taken from various angles, corresponding pixels do not necessarily depict the same location on the surface of an object. Specular materials lead to similar problems, since

bright highlights move on object surfaces when observed from different perspectives. Classical 3D reconstruction, i.e., triangulation of viewing rays, will fail to reconstruct consistent depth in these cases. It is also possible that the correct matches will not be obtained at all, if the search for corresponding pixels is strictly confined to the epipolar line.

Image Morphing does not suffer from any of the above mentioned constraints. As long as a perceptually valid bipartite matching of image locations is found, a convincing view interpolation can be synthesized [179]. The reconstruction is also not influenced by inaccurate camera calibration or dynamic scene elements. The downside is that virtual views are usually confined by the convex hull of the recorded images, i.e., no extrapolation of camera positions is possible. In cases where these locations are embedded into a lower-dimensional subspace, view interpolation is only possible in one or two dimensions. Another drawback for morphing-based algorithms is occlusion handling which cannot be solved by simple depth comparison. Furthermore, correspondence estimation is a 2D search problem and poses a less-constrained and therefore more difficult problem then a 1D search along epipolar lines.

5.2 A Hybrid Scene Model

In this section, I describe the hybrid approach of depth- and correspondence-based rendering that can cope with the specific limitations of the two separate techniques. In order to robustly handle real-world scenes I specify the iterative scheme for depth and correspondence estimation in Section 5.3. Technical details regarding the rendering will be discussed afterwards in Section 5.4.

Again, two real-world images A, B serve as input data. Camera parameters of A and B as well as the depth of every pixel are assumed to be known. For a given pixel location $\mathbf{x}_i$, this location in image A can be transformed to its world space position X_i using the DIBR equation (5.1), camera calibration and the reconstructed depth. As discussed above (Section 5.1), the backprojected location in B might not depict the same object due to several reasons:

- Images/pixels not captured at the same time instant and scene is dynamic
- Non-Lambertian reflectance

- Camera calibration errors
- Inaccurate depth estimate
- Other model violations (insufficient camera model, matching ambiguities, etc.)

The goal is to accurately project the pixel at position $\mathbf{x}_i$ in image A to its corresponding position $\mathbf{x}_j$ in image B, where $\mathbf{x}_j = \mathbf{x}_i + \mathbf{w}_{AB}(\mathbf{x}_i)$, cf. Eq. 5.4. The color values of the corresponding points should be visually similar:

$$I_A(\mathbf{x}_i) \sim I_B(\mathbf{x}_j) \tag{5.5}$$

Assuming knowledge of camera calibration, $\mathbf{x}_j$ can be deducted from its world space position X_j:

$$\mathbf{x}_j = \mathbf{C}_B(X_j) = \mathbf{C}_B(X_i + (X_j - X_i)) \tag{5.6}$$

For the sake of readability, world space correspondence maps $\mathbf{W}_{AB}(\mathbf{x}_i) = X_j - X_i = {\mathbf{C}_B}^{-1}(\mathbf{x}_j) - {\mathbf{C}_A}^{-1}(\mathbf{x}_i) = {\mathbf{C}_B}^{-1}(\mathbf{x}_i + \mathbf{w}_{AB}(\mathbf{x}_i)) - {\mathbf{C}_A}^{-1}(\mathbf{x}_i)$ are introduced. Further substitution reveals:

$$\mathbf{x}_j = \mathbf{C}_B({\mathbf{C}_A}^{-1}(\mathbf{x}_i) + \mathbf{W}_{AB}(\mathbf{x}_i)) \tag{5.7}$$

This means that by knowing depth, camera calibration and image correspondences, a valid transformation of every location in A to the image plane of B can be obtained. In contrast to DIBR, the errors mentioned above can be eliminated as long as visually plausible correspondences $\mathbf{W}_{AB}$ are known. Similar to warping-based rendering, intermediate views V can be rendered by interpolating corresponding positions. Instead of warping pixel location $\mathbf{x}_i$ towards $\mathbf{x}_j$ in image space, pixel locations are transformed to world space positions X_i, X_j and the final pixel location is determined by reprojecting the interpolated position to the image plane of V. While warping-based rendering only allows linear transitions between A and B, the proposed scheme allows to reproject the images to an arbitrary virtual camera V:

$$\begin{aligned} \tilde{I}_A(\mathbf{C}_V(X_i + b\mathbf{W}_{AB}(\mathbf{x}_i))) = I_A(\mathbf{x}_i) \sim \\ \tilde{I}_B(\mathbf{C}_V(X_j + a\mathbf{W}_{BA}(\mathbf{x}_j))) = I_B(\mathbf{x}_j),\ a + b = 1 \end{aligned} \tag{5.8}$$

Practically, this means that every pixel $\mathbf{x}_i$ in A is projected to its world space position X_i. According to the weight b, it is moved towards the corresponding world space

point X_j of image B. All pixels of image A are projected into the image plane of V, resulting in a warped image $\tilde{I}_A$. This process is repeated for image B, resulting in the intermediate output $\tilde{I}_B$. The final result is a blend of the two reprojected images:

$$I_V(\mathbf{x}) = a\,\tilde{I}_A(\mathbf{x}) + b\,\tilde{I}_B(\mathbf{x}) \tag{5.9}$$

This approach combines strengths of both approaches: Like DIBR, geometric information (depth) is used to project image texture into world space and to resolve occlusion. Similar to image morphing, transitions between different source images are possible by forward warping of pixels according to image correspondences. To make this hybrid approach possible, these transitions are applied in world space instead of image space. In contrast to DIBR, a plausible reprojection is possible despite imperfect depth reconstruction. This extends to the case where no depth can be reconstructed in parts of the image. In Section 5.3 I will explain how it works with a rough depth estimate. In contrast to warping-based rendering, the camera view can be extrapolated and depth information is used to better handle occlusion. Also, a view-dependent surface model is created as a by-product that is valuable for further applications such as re-lighting, shadow casting, and deep compositing.

5.3 Hybrid Reconstruction

In this section I will describe how an initial set of image correspondences is obtained, dense depth maps are derived from these correspondences and how the geometric information is used to compute a second generation of correspondences, Fig. 5.3.

5.3.1 SfM calibration

Using established SfM techniques, camera calibration [173] and (optionally) a sparse 3D point cloud representation of the scene [50] are estimated, Fig. 5.4. For every pixel $\mathbf{x}_i$ in a reference image A, the 3D scene structure is queried for a point X_i that projects to it: $|\mathbf{x}_i - \mathbf{C}_A(X_i)|_\infty < 0.5$ px. If such a point exists, a correspondence vector estimate $\tilde{\mathbf{w}}_{AB}(\mathbf{x}_i)$ is derived for any neighboring view B:

$$\tilde{\mathbf{w}}_{AB}(\mathbf{x}_i) = \mathbf{C}_B(X_i) - \mathbf{x}_i \tag{5.10}$$

Figure 5.4: Reconstruction of Tree18 [32]: **SfM calibration** and initial depth. First, **SfM** is used to calibrate the input images. Single points X_i of the sparse 3D point cloud are reprojected to the image planes of A, B, C to compute initial correspondences $\tilde{\mathbf{w}}_{AB}, \tilde{\mathbf{w}}_{AC}$.

5.3.2 Correspondence Estimation

The initial geometric reconstruction can be exploited to constrain the correspondence search. For the reasons mentioned in Section 5.2, $\tilde{\mathbf{w}}_{AB}(\mathbf{x}_i)$ should differ from $\mathbf{w}_{AB}(\mathbf{x}_i)$, but for the set of points reconstructed in Section 5.3.1, it can be assumed that the difference is small: $\tilde{\mathbf{w}}_{AB}(\mathbf{x}_i) \sim \mathbf{w}_{AB}(\mathbf{x}_i)$, Fig. 5.5. The correspondence search is based upon a optimization scheme (presented in Chapter 3) that minimizes the energy term $E = E_{data} + E_{smooth} + E_{symmetry}$. An additional geometric constraint E_{geom} is added that enforces consistency between the (sparse) geometric information and the estimated correspondences:

$$E(\mathbf{w}_{AB}) = E_{data} + E_{smooth} + E_{symmetry} + E_{geom} \tag{5.11}$$

$$= \sum_{\mathbf{x}} \|d_A(\mathbf{x}) - d_B(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_1 \tag{5.12}$$

$$+ \sum_{(\mathbf{x},\mathbf{y})\in\epsilon} \min(\alpha\|\mathbf{w}_{AB}(\mathbf{x}) - \mathbf{w}_{AB}(\mathbf{y})\|_1, d) \tag{5.13}$$

$$+ \sum_{\mathbf{x}} \min(\alpha\|\mathbf{w}_{AB}(\mathbf{x}) + \mathbf{w}_{BA}(\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x}))\|_2, d) \tag{5.14}$$

$$+ \sum_{\mathbf{x}} \min(\tilde{\alpha}\|\mathbf{w}_{AB}(\mathbf{x}) - \tilde{\mathbf{w}}_{AB}(\mathbf{x})\|_2, d) \tag{5.15}$$

As proposed by Liu et al. [115], matched pixels should have a similar appearance. This is expressed by the data term E_{data} (5.12), where $d_A(\mathbf{x})$ is the (multi-dimensional)

descriptor of a pixel location $\mathbf{x}$ in image A. The smoothness term E_{smooth}(5.13) enforces that all pixels $(\mathbf{x}, \mathbf{y})$ of a neighborhood ϵ have a similar correspondence vector. The bidirectional symmetry term $E_{symmetry}$ (5.14) for joint estimation of $\mathbf{w}_{AB}$ and $\mathbf{w}_{BA}$ is used as described in Chapter 3: Both correspondence fields $\mathbf{w}_{AB}$, $\mathbf{w}_{BA}$ are computed simultaneously. After each iteration of the global optimization, the intermediate result for $\mathbf{w}_{AB}$ is updated and used to evaluate the symmetric term of $\mathbf{w}_{BA}$ (and vice versa). E_{geom} (5.15) is the newly created geometric term that enforces $\mathbf{w}_{AB}$ to coincide with the geometric reprojection $\tilde{\mathbf{w}}_{AB}$ according to the depth model. The weighting parameter $\tilde{\alpha}$ is set to $\tilde{\alpha} = \alpha$ for every pixel where depth information is available and $\tilde{\alpha} = 0$ elsewhere.

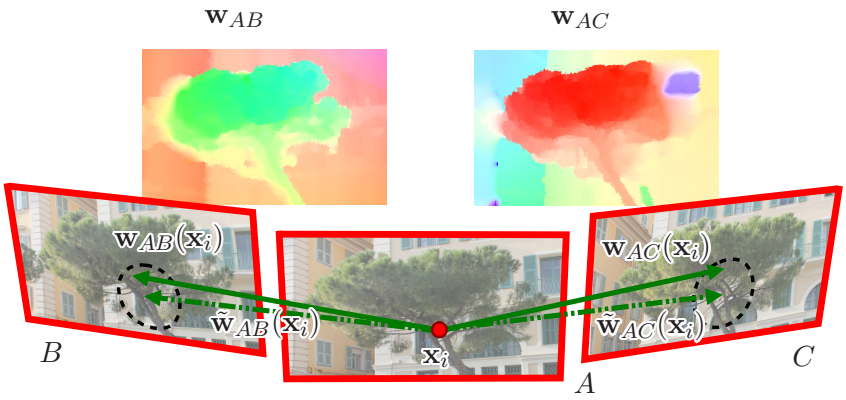


Figure 5.5: Reconstruction of Tree18 [32]. **Correspondence estimation** between neighboring images is performed. Correspondences $\mathbf{w}_{AB}$ are enforced to be similar to the initial correspondences $\tilde{\mathbf{w}}_{AB}$ derived from the geometric model.

5.3.3 Dense Depth Reconstruction

An actual world space position $\tilde{X}_i$ is obtained if at least one valid corresponding pixel $\mathbf{x}_j$ in another image B is known. The approximate world space position $\tilde{X}_i$ is obtained by triangulating the viewing rays, Fig. 5.6. Correspondences are defined as symmetric, if $\|\mathbf{w}_{AB}(\mathbf{x}_i) + \mathbf{w}_{BA}(\mathbf{x}_i + \mathbf{w}_{AB}(\mathbf{x}_i))\|_2 < e_{sym}$, where typically $e_{sym} = 3$ px. Due to the many possible errors in depth estimation, corresponding viewing rays cannot be expected to intersect in 3D space. Instead, $\tilde{X}_i$ is approximated as the point closest

to all viewing rays in a least-squares sense. The depth associated with $\mathbf{x}_i$ is the z-component of $\tilde{X}_i$ projected into image space coordinates of A. The reprojection error $e_{repr}(\mathbf{x}_i)$, i.e., the image space distance between $\mathbf{x}_i$ and $\mathbf{C}_A(\tilde{X}_i) = \tilde{\mathbf{x}}_i$, indicates if the reconstructed depth is valid. In order to deal with imprecise calibration, inaccurate correspondences and asynchronous captures, the reprojection error threshold is set to a high value of $e_{max} = 10$ px.

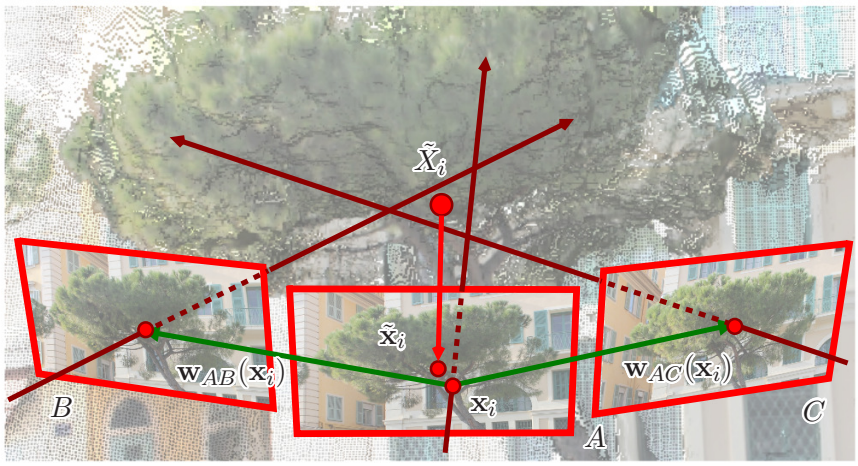


Figure 5.6: Reconstruction of Tree18 [32]. **Dense depth reconstruction** is based on triangulation of corresponding points. $\tilde{X}_i$ is approximated as the point with the minimal squared distance to all rays.

5.3.4 Depth Filtering

Since depth maps typically contain holes (i.e., invalid pixels) and outliers (due to the high tolerance to reprojection error), strong filtering is used to obtain a robust result: The (depth) images are assumed to consist of piecewise-planar surfaces. Therefore, each image is segmented into visually homogeneous regions. For this purpose, SLIC segmentation [1] is used. SLIC is configured to obtain 1000 segments for image sizes between one and two megapixels. For each superpixel, RANSAC is used to fit a plane to the reconstructed world space points, Fig. 5.7. Only points which are considered as valid contribute. For super-pixels without a sufficient number of valid pixels, depth is simply assigned by averaging neighboring segments. At least 5% of a segment's pixels should be valid, this threshold has been determined experimentally.

5.3.5 Refined Correspondence Estimation

After obtaining dense depth maps for any image A, correspondences are re-estimated once with updated geometric constraints $\tilde{\mathbf{w}}_{AB}$. During this second iteration of correspondence estimation, the weighting term $\tilde{\alpha}$ is re-evaluated based on the confidence in the depth reconstruction: $\tilde{\alpha} = \alpha \cdot \max(e_{max} - e_{repr}(\mathbf{x}), 0)/e_{max}$. It only affects optimization if $e_{repr}(\mathbf{x}) < e_{max}$. Again, the reason for the introduction of the geometric constraint is to steer the correspondence search towards geometrically plausible solutions. Where the initially computed correspondence maps result in a plausible geometric proxy (i.e., $e_{repr} < e_{max}$), the correspondence search is influenced by the depth information. In regions where they do not agree on a valid model, the correspondence search remains unconstrained. Effectively, this scheme constitutes a hybrid reconstruction: Whenever plausible geometric information can be derived, the correspondence search is forced to be consistent with it. Otherwise, a geometrically unconstrained solution is accepted.

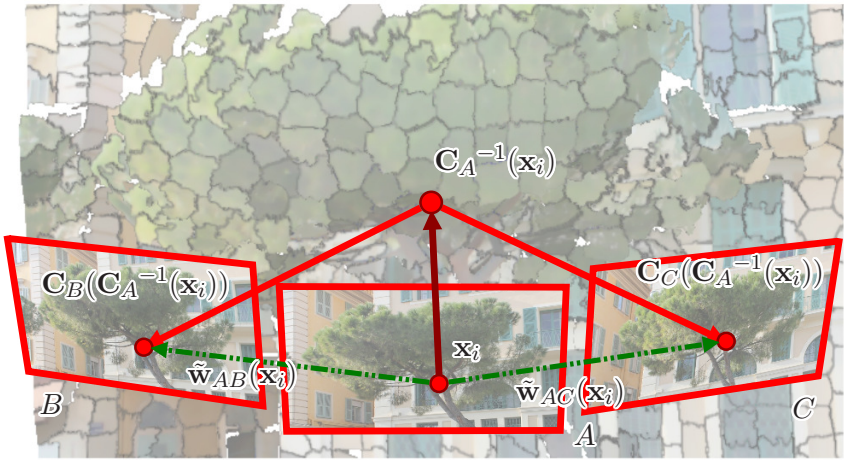


Figure 5.7: Reconstruction of Tree18 [32]. **Dense Depth filtering** is applied via plane fitting on each 2D SLIC[1] superpixel. The new depth values are used to update the geometric constraints $\tilde{\mathbf{w}}_{AB}$, which are used for a second iteration of correspondence estimation.

By employing both a symmetric (5.14) and a geometric term (5.15) simultaneously, consistency across the whole data set is encouraged. For scenes with large depth discontinuities, a segmentation of the foreground object(s) can also be taken into account. The two different layers are processed independently to prevent mismatches between fore- and background. Matching costs between pixels of different layers are set to the

same default (high) value that also applies to potential out-of-image correspondences, as proposed by Liu et al. [115].

5.4 Rendering

A GPU-based multi-image warping and blending scheme is used for rendering. Input data are a set of (calibrated) images as well as a weighting function that determines for every virtual view V a subset $S = \{A_1, A_2, ..., A_n\}$ of all recorded images and a set of weights $T = \{a_1, a_2, ..., a_n\}$, $|S| = |T| = n$. As a weighting scheme, the per-camera weighting of Pulli et al. [142] is used. Depending on the camera setup, n is set to $n \in \{2, 3, 4\}$. Images A_i, A_j that are contained in at least one common subset S are referred to as neighbors. Also, depth maps D_{A_i} and the correspondence maps $\mathbf{w}_{A_iA_j}$ between neighboring images are used.

World space correspondence maps $\mathbf{W}_{A_iA_j} = X_{A_j} - X_{A_i} = \mathbf{C}_{A_j}^{-1}(\mathbf{x} + \mathbf{w}_{A_iA_j}(\mathbf{x})) - \mathbf{C}_{A_i}^{-1}(\mathbf{x})$ are computed. The correspondence maps $\mathbf{W}_{A_iA_j}$ are only considered valid in regions where $\mathbf{w}_{A_iA_j}$ is symmetric. World space correspondences are propagated to invalid regions by using anisotropic diffusion [138] with two modifications: First, only regions with invalid values are updated in each iteration. Second, the depth value $D_A(\mathbf{x})$ is used for the evaluation of the nearest-neighbor difference. This ensures propagation only to spatially connected objects.

A dense grid of quads is rendered where every vertex represents a pixel location $\mathbf{x}$ in image A_i. Using the depth information for A_i, the world space coordinate X is calculated for a given vertex $\mathbf{x}$. According to $\mathbf{W}_{A_iA_j}$, it is warped towards its corresponding world space position in A_j by weight a_j. This warping step is repeated for every image $A_k \in S$ with weight a_k. The final world space location is reprojected onto the image plane of V. In order to handle disocclusions, e.g., areas where two objects seem to move apart, single mesh quads are discarded when two neighboring vertices $\mathbf{x}_i, \mathbf{x}_k$ differ too much in their depth values: $max(\mathbf{x}_k.d/\mathbf{x}_l.d, \mathbf{x}_l.d/\mathbf{x}_k.d) > e_{depth}$, typically $e_{depth} = 1.1$. These operations are performed in a GLSL geometry shader. This procedure is repeated for all images $A_j \in S, A_j \neq A_i$ and n different projections are obtained. When blending the n different reprojections, a soft z-Buffer as well as inpainting technique as described by Zheng et al. [208] are used.

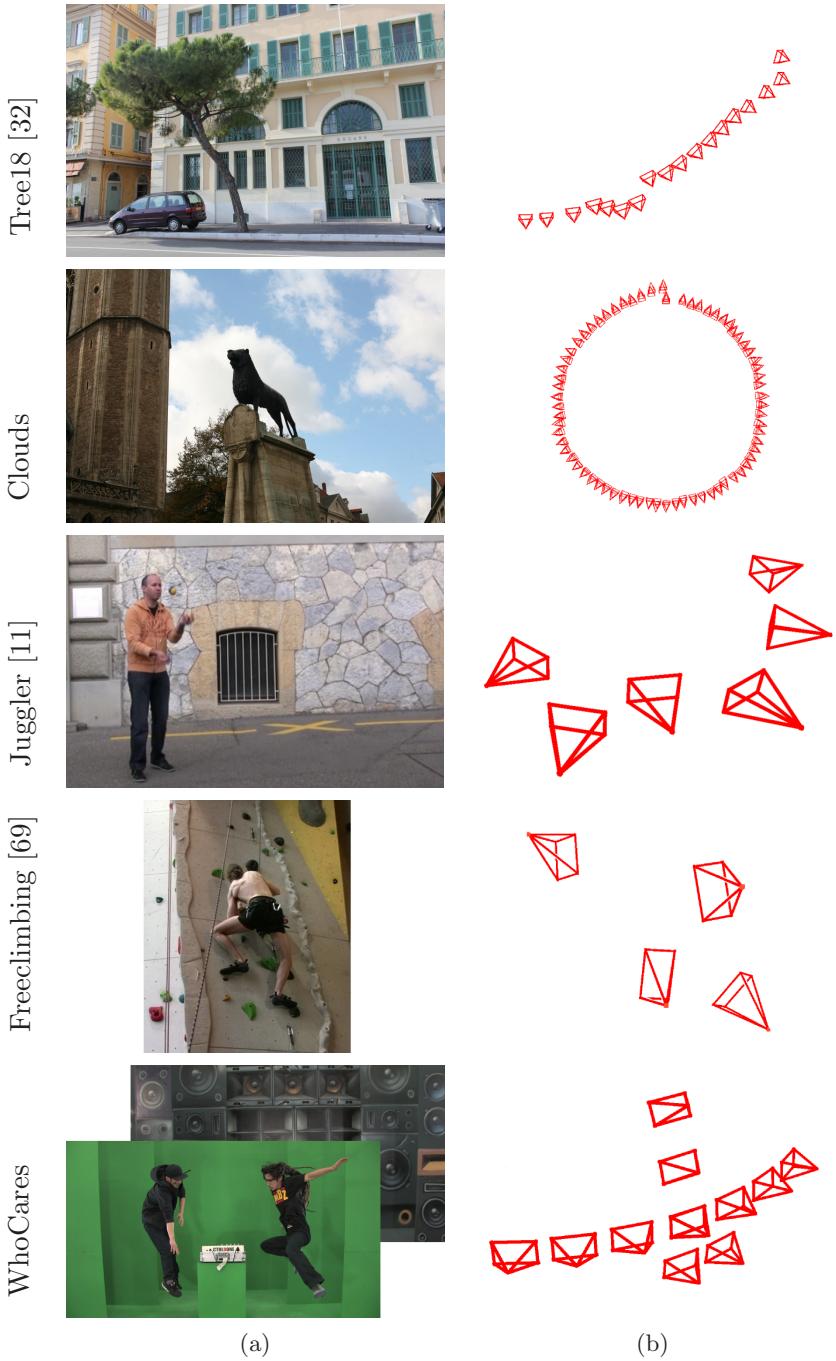
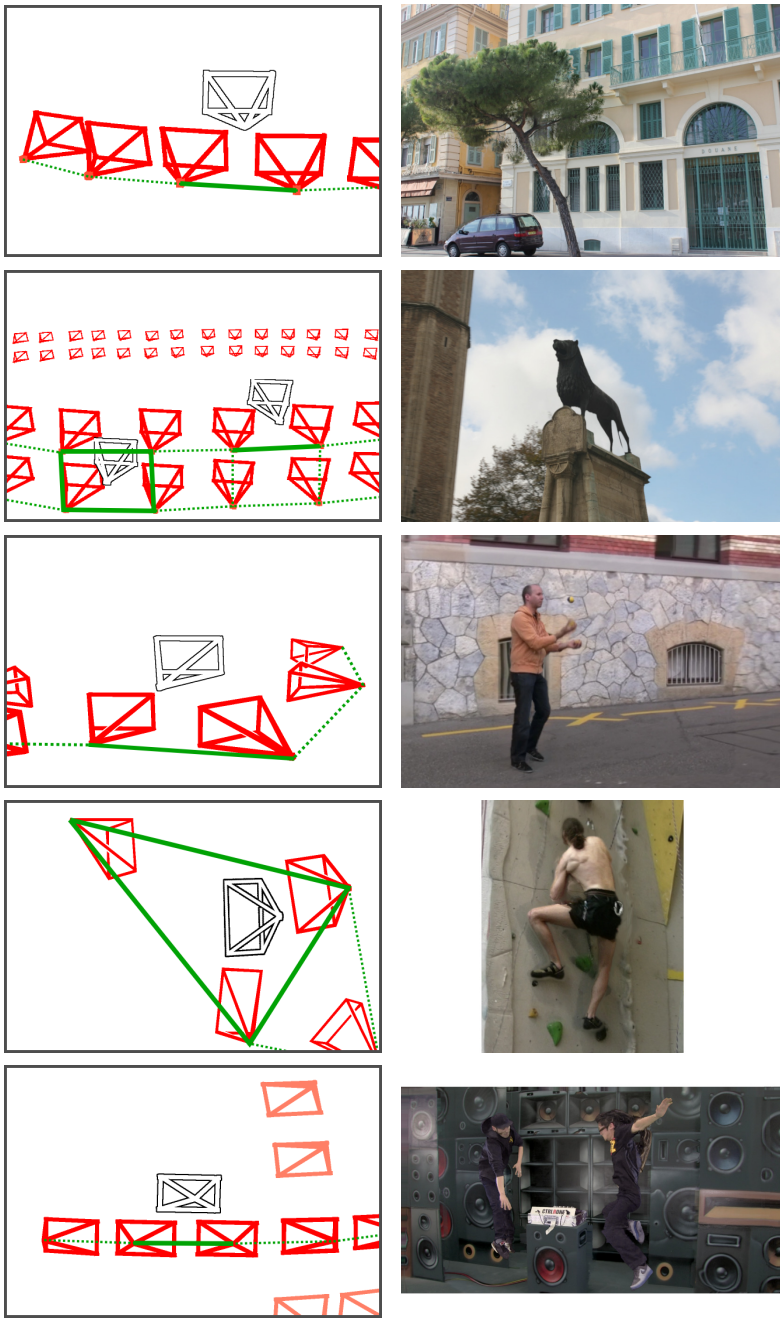


Figure 5.8: Overview of processed scenes. Scenes from top to bottom: **Tree18** [32] data set featuring 18 images of a single camera in an arc-like setup. **Clouds** contains 140 images captured with a single DSLR camera in a full circle around the central statue. **Juggler** [11] and **Freeclimbing** [69] contain images captured by 5 and 4 camcorders, respectively. **WhoCares** was captured using 11 camcorders. Visualizations from left to right: (a) shows an original image for each scene. (b) gives an overview of the camera setup.



(c) (d)

Figure 5.8: (cont'd) (c) describes which weighting scheme is used: a two-camera weighting scheme is used for Tree18, Juggler and WhoCares. In the Freeclimbing scene, each virtual view is rendered using three of the four original cameras. In the Clouds scene, the grid-like setup allows a bilinear interpolation of the four surrounding original images. (d) depicts images rendered with the proposed approach.

Figure 5.9: Comparison of state of the art (left) and the proposed hybrid approach (right), top to bottom: **Tree18**: In contrast to other depth-based approaches [32], the proposed hybrid approach aligns the semi-transparent car windows. **Clouds**: Close up of panorama rendering. State-of-the art panorama stitching [20] displays ghosting artifacts for moving scene contents. The proposed approach successfully aligns the clouds. **Juggler**: Cross-blending of billboards may result in ghosting artifacts [11], the piecewise planar depth and warping-based alignment remedy this effect.

Figure 5.9: (cont'd) Comparison of state of the art (left) and the proposed hybrid approach (right), top to bottom: **Freeclimbing**: A blend of two consecutive frames of a billboard-based rendering is compared to the hybrid approach. Despite the wide baseline, the free-climber can be aligned. **WhoCares**: For the production of the "WhoCares" music video, manual retouching of correspondences and rendered frames was required, Section 4.7. The proposed hybrid approach achieves comparable quality without user interaction.

5.5 Results and Evaluation

In order to give an understanding of the reconstruction and rendering workflow, I present and discuss results for five test scenes and highlight the separate challenges I encountered, Fig. 5.9. Free viewpoint renderings are presented for all scenes and various additional applications are showcased, i.e., image stabilization, panorama generation, (deep) compositing and relighting. To assess the visual quality of the proposed approach, online video results can be accessed at the project website [34]. Four of the scenes are publicly available data sets that have been used by other state-of-the-art free viewpoint algorithms. By directly comparing against these very different state-of-the-art algorithms, the generality of the proposed approach is demonstrated. All scenes show dynamic content: None of the data sets feature synchronous image captures, they have either been recorded by a single moving DSLR camera or several non-genlocked camcorders. All scenes feature image regions that do violate the epipolar constraint to some extent. This manifests in a high reprojection error of non-static objects, i.e., treetops (Tree18, Clouds), clouds (Clouds) and moving actors (Juggler, Freeclimbing, WhoCares).

Tree18 dataset All 18 photographs of the Tree18 dataset provided by [32] are used. They are downsampled to 1296×864 px before further processing. Due to the arc-like setup, n is set to $n = 2$, cf. Section 5.4, so that the two closest original views are used to render an in-between view. The sparse depth information (obtained by the multi-view stereo algorithm of Furukawa et al. [50]) is used to constrain the initial correspondence map estimation. In the video comparison, the proposed approach can compete with the state of the art [32]. While the rendering was created without any additional input, Chaurasia et al. allow the user to give cues for edges and depth. The moving treetop and the semi-reflective car windows violate basic assumptions of DIBR and depth is challenging to reconstruct, as reported by Chaurasia et al. [32]. Since the (unconstrained) correspondence search yields plausible results for these images regions, the hybrid approach can produce visually pleasing renderings, Fig. 5.9.

Cloud dataset Images were captured with a DSLR and processed at 1620×1080 px. The scene is segmented into fore- and background with Nuke rotoscoping tools [181]. Both layers are processed and rendered independently. I took 140 photos of the scene,

half of them were taken in a standing and the other half in a crouching position. They span a full circle around the lion statue in the center. Two images from the upper row and two images from the lower row contribute to one virtual view ($n = 4$). Although the scene seems easy to process due to the high density of views, the dynamic sky poses a hard challenge: Since the clouds had been moving during the ten-minute capturing process, no valid depth can be reconstructed. Zimmer et al. [209] report that only an unconstrained optical flow is able to cope with this kind of scene elements. Using unconstrained correspondence search for these areas, it is possible to align the clouds in the final render.

Juggler dataset Six handheld camcorders (960×544 px) were used to capture this sequence. the unstructured video-based rendering application by Ballan et al. [11] is used to create a short time-freeze free viewpoint video clip. I recreate the camera motion with the proposed hybrid approach. Similar to the Tree18 scene, n is set to $n = 2$. As input, the publicly available segmentation data and background reconstruction are used. In contrast to the billboard-switching approach of Ballan et al., the proposed hybrid approach provides a consistently deforming actor, Fig. 5.9. While their interactive viewer focuses on transitions between two neighboring cameras, a single continuous camera arc can be rendered with CDIBR. I would like to refer to the online video resources [34] for a direct comparison.

Freeclimbing dataset The biggest challenge on this data set is the wide camera baseline. The four camcorders (1920×1080 px) are spaced $> 40\,^\circ$ apart. Due to the rhomb-shaped setup, n is set to $n = 3$, so that three of the four original images contribute to the final synthetic render. Ballan et al. [11] use a foreground segmentation and a billboard approximation of the freeclimber for free viewpoint rendering. Similar to their approach, a foreground segmentation is used and both layers are processed independently. This enables setting different parameters to the foreground, i.e., $e_{max} = 20$ px. Otherwise, no valid geometry can be obtained. In order to assess the contribution of the hybrid approach, three renders of this scene are produced. First, a single fronto-parallel billboard is used for the actor, Fig. 5.9 (left). Second, a per-pixel depth value is obtained with the proposed reconstruction pipeline and DIBR is

used for rendering (cf. online results video [34]). Third, the full approach of hybrid reconstruction and rendering is employed, Fig. 5.9 (right).

WhoCares dataset Since this scene was captured in a green-screen environment, chroma-keying allows for unsupervised foreground segmentation. Eleven unsynchronized camcorders (1920×1080 px) were used for the capture, several dozen images of each camera are processed.

I recreate a five-second sequence of the "WhoCares" music video presented in Chapter 4 with the novel hybrid approach. Since the camera moves in an arc along the original camera positions, n is set to $n = 2$. In order to achieve convincing results, I previously relied on elaborate tools for correspondence map correction [224], manual retouching with GIMP [183] and manual compositing of the different layers. For comparable quality, the hybrid approach does not require any manual retouching of the rendered data, Fig. 5.9.

5.5.1 Limitations

Additional input data helps to increase the robustness. For three of the five shown scenes (Freeclimbing, Clouds, Juggler), an additional manual foreground segmentation helps to obtain valid correspondences and depth values. I argue that image segmentation is a more common and simple task compared to depth [32] or correspondence correction [224]. Still, a fully automatic processing would be preferable. A similar problem is the automatic SLIC super-pixel segmentation of the input images. Unlike correspondences or depth, the initial segmentation is not refined during the iterative scheme. This results in spurious rendering artifacts, e.g., popping. Promising research has been conducted that aims to find a global solution to depth and segmentation simultaneously [26] and could be incorporated into the proposed approach.

Although the approach can cope with a sparse camera placement as shown in the freeclimbing scene, it is not feasible for all setups. E.g., the Rothman scene from [11] features cameras that are spaced too widely apart to allow for a faithful reconstruction using image correspondences. The placement of virtual cameras is also limited to positions close to the actual setup. In the most extreme case (Freeclimbing scene) distances of 40° can be bridged, so one would need at least nine cameras to render a full 360° path. For such sparse camera placements other free viewpoint approaches typically

require studio setups along with highly accurate camera calibration and synchronization [175].

5.6 Applications

Instead of devising my own free viewpoint editing framework, the renderer is used as a backend for existing tools. After the reconstruction phase, cf. Section 5.3, a simple 3D proxy geometry of the scene is exported along with the camera positions. Scene exporters to compositing and 3D modeling software (Nuke [181] and Blender [17]) have been implemented for this purpose. Camera paths can be created using standard 3D animation tools. The final camera path is exported to the renderer. Different applications are conceivable with the hybrid approach that are difficult or impossible to recreate with other techniques:

(a) (b)

Figure 5.10: Additional applications. The cloud data set (a) is re-rendered with novel views placed on a perfect ellipse (b). By pointing the viewing rays to the scene center and synthesizing motion blur, a "super-stabilized" virtual camera is created. This is visualized by tracking a church window (red) in the background and the nose of the foreground (green) statue across multiple frames.

Image Stabilization A "super-stabilized" camera trajectory can be rendered for the clouds data set. The virtual camera is placed on a perfect ellipse around the central statue and the image sequence is re-rendered with the virtual camera constantly pointing towards the central statue. In order to visualize the steady movement of objects in image space, the tracking results for two objects in the scene are plotted, Fig. 5.10. For the original 70 frame long sequence, 3500 synthetic views are rendered. Their camera projection centers are equally spaced along the ellipse. 25 synthetic

frames are averaged to simulate a 1/50s shutter. I would like to point out that by using only correspondence-based view interpolation (i.e., the purely image warping-based technique presented in Chapter 4), this kind of view extrapolation is not possible.

Panorama Creation Since a believable 360 ° rendering of the background is achieved, a panoramic representation of the scene can be created by combining small (3-pixel wide) slices of the 3500 output images. This results is a 10500 pixel-wide panorama of the background, Fig. 5.11. Panorama stitching algorithms often assume that both camera position and scene content remain static during acquisition [20]. Therefore, the rendered panorama often displays ghosting artifacts when these assumptions are violated, Fig. 5.9.

Figure 5.11: Additional applications: 360 ° panorama creation. By concatenating 3 pixel-wide slices from a complete 360 ° rendering of the background, a panoramic view of the whole scene can be rendered. All dynamics scene elements (trees, clouds) are aligned in a visually plausible way. For the sake of readability, the full 360 ° panorama is separated into two 180 ° images.

Compositing The camcorder recording of a graffiti-covered brick wall is augmented with a free viewpoint rendering of the "WhoCares" foreground layer. The movement of the handheld camera is tracked using matchmoving software (Cameratracker [181]), Fig. 5.12 (a). In Blender [17], the handheld camera motion is aligned with proxy geometry from the "WhoCares" scene by allowing the user to define a translation, rotation and scale factor, Fig. 5.12 (b). Additionally, 3D objects and contact shadows are inserted into the scene, Fig. 5.12 (c). The 3D objects are created with standard modeling and texturing tools available in Blender. Soft contact shadows are created with the

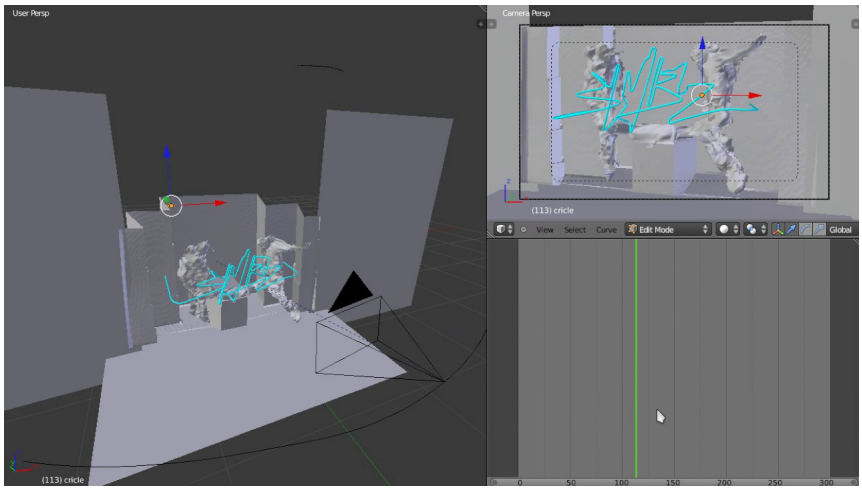
proxy geometry as a shadow caster and a planar approximation of the brick wall as a shadow receiver. After a pass of the 3D box and the shadow layer is rendered, camera matrices are exported to the free viewpoint renderer. The four different render layers (i.e., handheld recordings, shadow layer, 3D box, free viewpoint video) are composited in Nuke. Since valid depth values for all render passes can be obtained, deep compositing can be used, i.e., the different layers are sorted automatically according to their depth values. For other, purely image-based approaches like CBR (Chapter 4) or [42]), the creation of this kind of effect would require additional efforts.

Re-lighting In Blender, the captured scene can be aligned with hand-modeled 3D objects. A 3D model of a neon light source is inserted into the "WhoCares" scene, Fig. 5.13 (a). A standard diffuse material is assigned to the proxy geometry and a pass of the geometry is rendered which is lit by the neon lighting tube, Fig. 5.13 (b). The original free viewpoint rendering, a rendering of the neon light source as well as the rendering of the diffusely lit background geometry are exported to Nuke for compositing. When simply merging the 3D object with the original free viewpoint material, the lighting of the scene seems odd, Fig. 5.13 (c). In order to achieve a convincing "neon"-look, the diffusely lit geometry is merged with the actual free viewpoint rendering, Fig. 5.13 (d). The shown effects (shadows, local lighting) can only be created because of the geometric data reconstructed by the hybrid approach.

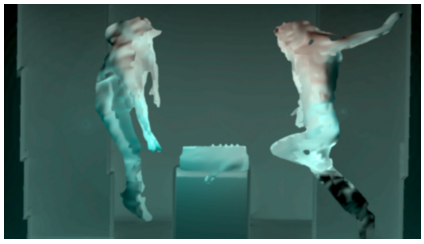
5.7 Summary

I present a versatile and robust hybrid formulation for free viewpoint rendering. The proposed approach can compensate for inaccurately estimated scene geometry by incorporating visually plausible correspondences into the rendering equation. I link depth and correspondence estimation by a soft geometric constraint in the iterative reconstruction pipeline. The versatility of the approach is demonstrated on a wide variety of established test scenes, as well as different applications. The results are compared with the state of the art, showing that the hybrid rendering scheme works without any additional user input (Tree18, WhoCares) or produces better results (Clouds, Juggler).

Figure 5.12: Additional applications: Free viewpoint compositing. (a) The camera movement of a freely moving camcorder filming a graffiti-covered wall is reconstructed with commercially available camera tracking software [181]. (b) The sparsely reconstructed point cloud is aligned with a proxy geometry of the "WhoCares" sequence and a hand-modeled 3D box in Blender [17]. (c) The aligned camera positions are exported to the free viewpoint renderer and the results are composited with the original footage. (d) Using a planar approximations for the graffiti wall, soft contact shadows for the 3D box and the free viewpoint proxy geometry can be rendered. (e) Similar to the shadow layer in the "WhoCares" music video, the shadows can be incorporated into the final composite.



(a)



(b) (c)

(d)

Figure 5.13: Additional applications: Free viewpoint relighting. (a) In Blender [17], a 3D neon tube is hand-modeled and inserted into a proxy representation of the scene. (b) The proxy geometry created by the hybrid reconstruction scene is linked to a standard Lambertian material and lit with the neon tube using the internal Blender renderer. (c) In Nuke, the original free viewpoint video is merged with the neon tube. (d) To create a convincing relighting effect, the original footage is merged with the diffusely lit proxy geometry.

6

Discussion and Conclusion

In this thesis I presented algorithms for processing and rendering image-based free viewpoint video. I showed in Chapter 3 that by combining state-of-the-art computer vision algorithms, automatic correspondences for high-resolution image pairs can be estimated. These correspondence maps allow visually plausible in-between renderings even for difficult scenes. In Chapter 4 I used these correspondence maps to devise a free viewpoint system that is based on multi-image interpolation, also referred to as correspondence-based rendering (CBR). Even for non-synchronized cameras placed in an uncontrolled outdoor environment, visually plausible in-between views can be rendered. Since a holistic approach is used to interpolate the view in both space and time, various popular cinematic effects such as time-freeze, slow motion and "bullet time" are possible. I proposed an extension for stereoscopic free viewpoint rendering. Furthermore, the versatility and robustness of the renderer were extensively tested in an actual production of a three-minute music video. In order to extend the possibilities of image-based free viewpoint video, I presented a rendering scheme that combines correspondence- and depth-image-based rendering (CDIBR) in a single framework in Chapter 5. Similar to the initial image-based renderer, visually plausible correspondence maps are the fundamental ingredient. By employing image warping in 3D space, the scene can be re-rendered from arbitrary camera location. The robustness is improved by iteratively refining scene geometry and pairwise image correspondences. I directly compared the approach to current state-of-the-art approaches in free viewpoint rendering. In order to document the progress made in view interpolation and to shed some light on the question which free viewpoint renderer is preferable in different ap-

plications, I will directly compare and discuss both approaches in Section 6.1 before pointing out possibilities for future work in Section 6.2 and conclude in Section 6.3.

6.1 Discussion

Although it might look like that CDIBR is the natural successor to CBR, each method has its own merits, and a preference towards one of them should always be based on the application at hand. I will look at the different aspects of the two approaches.

6.1.1 View Extrapolation and Stereoscopic Rendering

In CBR, only view interpolation is possible in the basic framework. Although this restriction is alleviated somewhat by the possibility to produce stereoscopic video via depth-image-based rendering, the possibilities to generate arbitrary camera views are still limited. One advantage of stereoscopic rendering via CBR is that it guarantees that no vertical disparities are visible.

View extrapolation with CDIBR is possible out-of-the-box. It is feasible as long as the virtual camera is reasonably close to the original views. The reconstructed depth values are possibly more robust because multiple correspondence maps are evaluated for their computation, and an additional filtering step is applied. Although stereoscopic rendering is possible without any further modifications, the absence of vertical disparities is currently not guaranteed. Since all 3D pixel positions are interpolated depending on the current blending weights, and because these weights depend on the virtual camera position, corresponding pixels may be visible at different 3D positions in the left and right virtual view. A stereoscopic rendering scheme could be devised which assigns the exact same blending weights for the left and right virtual view in order to effectively prevent vertical disparities.

6.1.2 Temporal interpolation

In CBR, I developed a holistic interpolation framework that can cope with non-synchronized captures and render in-between views in the temporal domain. This allows for arbitrary space-time paths of the virtual camera.

In CDIBR, I currently only consider in-between views for a given time instant. Although a temporal interpolation scheme could be devised that is similar to the one

used for CBR, experimental tests have yet to be conducted. An interesting extension to depth estimation could be to also explicitly reconstruct the 3D velocity of each surface point. In fact, this would be complementary to the scene flow reconstruction of Klose et al. [221] who computes depth, velocity and surface orientation from multiple pairwise correspondences.

6.1.3 User Correction

Although the processing pipeline can automatically process arbitrary multi-view captures, manual corrections by a user often result in a considerable increase in rendering quality. Is is especially true for difficult scenes with occlusions, large depth discontinuities and wide camera baselines. Corrections in CBR are currently performed with custom-made tools ([224, 247]) that allow local corrections of the correspondence information.

In CDIBR, I avoided relying on manual corrections by the user. For scenes with large depth discontinuities, a scene segmentation into fore- and background proved to be sufficient to allow for correct correspondence estimation. Still, scenes may exist where the automatic correspondence estimation will fail. For these failure cases, corrections of the processed data can possibly be incorporated into different stages of the image processing pipeline. Since two iterations of correspondence estimation are performed, user corrections can be incorporated into either one or both iterations. Similar to the corrections proposed by Chaurasia et al. [32], corrections of the depth maps are also conceivable. According to my experience, however, it is quite hard to manually assign depth values to image regions, since accurate depth is usually hard to estimate for a human operator. Alternatively, the CDIBR reconstruction pipeline offers the possibility to include depth data from arbitrary sources. For example, the depth data captured by the Microsoft Kinect could be used to create correspondence priors.

6.1.4 Authoring Tools

For CBR, a custom authoring interface is built to control the virtual view parameters. This provides a very fast and direct way to edit the virtual camera trajectory. Only three camera parameters θ, ϕ, t have to be defined for the virtual camera and immediate previews can be inspected at real-time rendering speed. As shown in an extension to the *Virtual Video Camera* project, a media player is available that also allows real-time

view changes and arbitrary playback rates [241, 242]. Both the authoring tool and the media player interface allow a simple and direct control of the virtual camera. However, they lack some more elaborate features, e.g., interactive compositing with other 2D and 3D content.

In CDIBR commercial 3D modeling and compositing tools, i.e., Nuke and Blender, were employed to author the path of the virtual camera. The main advantage is that users can make use of the elaborate modeling, rendering and compositing tools at their disposal. The virtual camera can be matched to the camera paths from other recordings, the view-dependent geometry can be used to create shadows and other interactions with the scene and deep compositing can automatically combine various layers of free viewpoint and 3D renderings. The downside is that a live preview is currently only available in low quality. The view-dependent geometry of a reference camera is used as a makeshift replacement of the actual free viewpoint rendering, which is only available after an off-line render pass. Further integration of the renderer into existing tools or more elaborate previews of the free viewpoint video will help to better estimate the final look of a free viewpoint sequence.

6.1.5 Computational Costs

The computational costs mainly depend on the correspondence estimation stage. Since CDIBR has two computation passes, and depth map computation is more elaborate, the overall computation time is at least doubled. Furthermore, correspondence maps that are not needed for the actual rendering may have to be computed for the sake of robust depth estimation.

6.1.6 Comparison

CDIBR provides more possibilities for free viewpoint rendering and potentially requires less user corrections during image processing. However, CBR proved to be an adequate rendering framework for an actual music-video production. When employing CDIBR to a similar task, several aspects of the pipeline will have to be re-evaluated. The current authoring interface builds upon the strengths of existing 2D/3D tools but does not yet provide real-time rendering feedback. Also, it is not clear yet at which point additional user corrections and manual clean-up of the data are most efficient.

6.2 Future Work

Many directions for future work exist. Correspondence and depth-image-based rendering (CDIBR) has yet to prove its applicability in the context of actual visual effects production. In this context, a more efficient correspondence estimation algorithm is necessary for processing the data in a more acceptable time frame. This is especially true when considering the demand for productions that use ever-more spatial and temporal resolution. Furthermore, a tight integration into 2D or 3D authoring software will help to develop visual effects more quickly and efficiently. The possibility to re-render the scene from arbitrary locations in conjunction with the availability of scene-dependent geometry also opens up novel research opportunities. The initial results of free viewpoint compositing and relighting (Chapter 5) are a mere appetizer for further in-depth research.

6.3 Conclusion

In this thesis, I presented an image-based rendering framework that builds on visually plausible correspondence estimation. An important lesson learned from the development and testing of the algorithms is that any given approach can only be properly evaluated within the context of an actual application. Although producing initial experimental results with publicly available data sets is a good way to test the feasibility of any approach, it is important to think of the challenges of the application at hand. The improvements introduced to the rendering framework in CDIBR were largely motivated by the shortcomings revealed during the "Who Cares" music video production. It proved to be tedious but necessary to go back between fundamental algorithm design and extensive testing on actual real-world data.

Symbols

(θ, ϕ)

Angular parameters: They are commonly used to encode the spherical coordinates of a point relative to the scene center. In the plenoptic function (Section 2.1), (θ, ϕ) encode the direction of a viewing ray.

(u, v)

Planar parameters: The vector (u, v) is used at several occasions for parametrization of 2D entities. Examples are the uv plane in light field rendering (Section 2.2.1) and for correspondence vectors (Section 2.1.3).

A, B, C, D

Views of a scene: A, B, C, D are used to represent a view of a scene that has been captured by an actual camera. A view A is associated with an image I_A, a recording time t_A, a projection center $\mathbf{p}_A$, a depth map D_A and a camera matrix $\mathbf{C}_A$.

I

Image: An image is a 2D structure that can be evaluated for a given pixel position $\mathbf{x}$. Depending on image content, a lookup in I may result in a one-dimensional scalar value $I(\mathbf{x}) = b$ (where b is commonly referred to as image brightness) or a three dimensional (color) vector $I(\mathbf{x}) = (r, g, b)$.

V

Virtual view of a scene view: V is used to represent a view of a scene that has not (necessarily) been captured by an actual camera. Commonly, an image I_V, a (virtual) recording time t_V, a projection center $\mathbf{p}_V$ and a camera matrix C_V are associated with a virtual view V.

X

Euclidean position in world space coordinates. A world space position X can be approximatively estimated for a given image space location $\mathbf{x}$ in a view A with camera matrix $\mathbf{C}_A$: ${\mathbf{C}_A}^{-1}(\mathbf{x}) = (X)$.

Ψ

Mapping from Euclidean space to navigation space $\mathcal{N}$: This function evaluates the extrinsic parameters of a recorded image and maps it to the three parameters (θ, ϕ, t) of the navigation space: $\Psi : (\mathbf{R}, \mathbf{p}, t) \mapsto (\theta, \phi, t)$.

α, c

Regularization parameters. The two parameters α, c are used to adjusted the weighting of the smoothness term (α) and the cutoff (c) for belief propagation optimization.

ϵ

Neighborhood of pixels. More formally, pixels $\mathbf{x}, \mathbf{y}$ are neighbors if the edge $(\mathbf{x}, \mathbf{y})$ is included in a set of edges ϵ. Typically, pixel neighboords are defined by vertical and horizontal neighbors in the image lattice (4-neighborhood).

λ

Wavelength of light.

C

Projection matrix: To each view A a projection matrix $\mathbf{C}_A$ is associated that encodes extrinsic and intrinsic parameters of a pinhole camera model. A given world space point X can be transformed into the image space location $\mathbf{x}$ of A with the operation: $\mathbf{x} = \mathbf{C}_A(X)$. The inverse operation is analogously define: ${\mathbf{C}_A}^{-1}(\mathbf{x}) = (X)$. For the sake of brevity, this operation (and its inverse) includes the perspective division: $\mathbf{C}_A(X) = \mathbf{C}_A\, X / (\mathbf{C}_A\, X.x, \mathbf{C}_A\, X.y, 1, 1)^T$.

R

Rotation (matrix) of an arbitrary view. Together with the projection center $\mathbf{p}$ and the intrinsic parameters, a camera projection matrix $\mathbf{C}$ can be obtained.

W

World space correspondence vector (or flow vector): The vector $\mathbf{W}_{AB}(\mathbf{x_i})$ expresses correspondence between pixel location $\mathbf{x}_i$ in image I_A and location $\mathbf{x}_j = \mathbf{x}_i + \mathbf{w}_{AB}(\mathbf{x}_i)$ in image I_B. Unlike the image space equivalent $\mathbf{w}_{AB}$, the correspondence is expressed in Euclidean world space position: It is the difference between the world space locations X_i and X_j: $\mathbf{W}_{AB}(\mathbf{x_i}) = X_j - X_i = {\mathbf{C}_B}^{-1}(\mathbf{x}_j) - {\mathbf{C}_A}^{-1}(\mathbf{x}_i)$.

p

Projection center of an arbitrary view. Together with the Rotation (matrix) $\mathbf{R}$ and the intrinsic parameters, a camera projection matrix $\mathbf{C}$ can be obtained. If $\mathbf{p}$ and $\mathbf{R}$ are known, the translation vector of $\mathbf{C}$ can easily be computed since $\mathbf{R}$ defines all entries of $\mathbf{C}$ except the tranlation vector and $\mathbf{C}\,\mathbf{p} = (0, 0, 0, 1)^T$.

v

Navigation space position. For view interpolation, all recorded images are mapped to vertices $\mathbf{v}$ in navigation space $\mathcal{N}$.

w

Correspondence vector (or flow vector): The vector $\mathbf{w}_{AB}(\mathbf{x})$ expresses correspondence between pixel location $\mathbf{x}$ in image I_A and location $\mathbf{x} + \mathbf{w}_{AB}(\mathbf{x})$ in image I_B.

x

Pixel position in image space. In the first part of the thesis, $\mathbf{x}$ is a 2D vector (Chapters 3,4). Later, it is used as a homogenous image space coordinate (Chapter 5).

$\mathcal{N}$

Navigation space. During correspondence-based rendering (Chapter 4), all recorded images are embedded into the so-called navigation space $\mathcal{N}$. This embedding serves two purposes: First, view interpolation can be done in a low-dimensional space. Second, the three dimensions (θ, ϕ, t) offer an intuitive navigation for the user.

τ

Tetrahedron. For view interpolation, all recorded images are mapped to vertices $\mathbf{v}$ in navigation space $\mathcal{N}$. A (constrained) Delaunay tessellation partitions this space into tetrahedra τ.

e

Error: Used as a threshold for several errors.

t

Time: a capture time t_A is associated with a view A. Note that t has a different meaning in the context of light field rendering (Section 2.2.1), where it is traditionally used in the 4-dimensional parametrization (u, v, s, t) of light fields.

Bibliography

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Ssstrunk, S. SLIC Superpixels. Tech. rep., EPFL, EPFL, 2010.

[2] Addison, A. C., MacLeod, D., Margolis, G., Hashoah, B., Naimark, M., and Schwarz, H.-P. Museums without walls (panel session): new media for new museums. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 480–481.

[3] Adelson, E. H., and Bergen, J. R. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing* (1991), MIT Press, pp. 3–20.

[4] Agin, G. J., and Binford, T. O. Computer description of curved objects. *IEEE Trans. Comput. 25*, 4 (Apr. 1976), 439–449.

[5] Agisoft. PhotoScan. `http://www.agisoft.ru/`, Nov. 2012.

[6] Alvarez, L., Deriche, R., Papadopoulo, T., and Sánchez, J. Symmetrical dense optical flow estimation with occlusions detection. *IJCV 75*, 3 (2007), 371–385.

[7] Autodesk. 123D Catch. `http://www.123dapp.com/catch`, Nov. 2012.

[8] Bai, X., and Sapiro, G. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV 82*, 2 (2009), 113–132.

[9] Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., and Szeliski, R. A database and evaluation methodology for optical flow. *International Journal of Computer Vision 92* (2011), 1–31.

[10] Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., and Szeliski, R. A database and evaluation methodology for optical flow. In *ICCV* (2007), pp. 1–8.

[11] Ballan, L., Brostow, G. J., Puwein, J., and Pollefeys, M. Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM Trans. on Graphics (Proc. SIGGRAPH) 29*, 3 (July 2010), 87:1–87:11.

[12] Battiti, R., Amaldi, E., and Koch, C. Computing optical flow across multiple scales: An adaptive coarse-to-fine strategy. *International Journal of Computer Vision 6* (1991), 133–145.

[13] Baumgart, B. G. *Geometric modeling for computer vision.* PhD thesis, Stanford, CA, USA, 1974. AAI7506806.

[14] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. Speeded-up robust features (surf). *Computer Vision and Image Understanding 110*, 3 (2008), 346 – 359. Similarity Matching in Computer Vision and Multimedia.

[15] Beeler, T., Hahn, F., Bradley, D., Bickel, B., Beardsley, P., Gotsman, C., Sumner, R. W., and Gross, M. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30* (August 2011), 75:1–75:10.

[16] Beier, T., and Neely, S. Feature-based image metamorphosis. *Computer Graphics (Proc. of SIGGRAPH'93) 26*, 2 (1992), 35–42.

[17] blenderinstitute. Blender website. `http://www.blender.org/`, July 2011.

[18] Boissonnat, J.-D. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph. 3*, 4 (Oct. 1984), 266–286.

[19] Bradley, D., Heidrich, W., Popa, T., and Sheffer, A. High resolution passive facial performance capture. *ACM Trans. Graph. 29*, 4 (July 2010), 41:1–41:10.

[20] Brown, M., and Lowe, D. G. Automatic panoramic image stitching using invariant features. *Int. J. Comput. Vision 74*, 1 (Aug. 2007), 59–73.

[21] Brown, M., Szeliski, R., and Winder, S. Multi-image matching using multi-scale oriented patches. pp. 510–517.

[22] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. High accuracy optical flow estimation based on a theory for warping. In *Computer Vision - ECCV 2004*, T. Pajdla and J. Matas, Eds., vol. 3024 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 25–36.

[23] Brox, T., and Malik, J. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33*, 3 (2011), 500–513.

[24] Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M. Unstructured Lumigraph Rendering. In *Proc. of ACM SIGGRAPH'01* (New York, 2001), ACM Press/ACM SIGGRAPH, pp. 425–432.

[25] Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 7 (2012), 1281–1298.

[26] Campbell, N. D., Vogiatzis, G., Hernández, C., and Cipolla, R. Automatic object segmentation from calibrated images. In *Visual Media Production (CVMP), 2011 Conference for* (Nov. 2011), pp. 126 –137.

[27] Carranza, J., Theobalt, C., Magnor, M. A., and Seidel, H.-P. Free-viewpoint video of human actors. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 569–577.

[28] Casas, D., Tejera, M., Guillemaut, J.-Y., and Hilton, A. 4d parametric motion graphs for interactive animation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2012), I3D '12, pp. 103–110.

[29] Cashman, T. J., and Fitzgibbon, A. W. What shape are dolphins? building 3d morphable models from 2d images. *IEEE Trans. Pattern Anal. Mach. Intell. 35*, 1 (Jan. 2013), 232–244.

[30] Catmull, E., and Rom, R. A class of local interpolating splines. *Computer aided geometric design 74* (1974), 317–326.

[31] Chambolle, A., and Pock, T. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Imaging Vis. 40*, 1 (May 2011), 120–145.

[32] Chaurasia, G., Sorkine, O., and Drettakis, G. Silhouette-Aware Warping for Image-Based Rendering. *Computer Graphics Forum 30*, 4 (2011), 1223–1232.

[33] Chen, S. E., and Williams, L. View interpolation for image synthesis. In *Proc. of ACM SIGGRAPH'93* (New York, 1993), ACM Press/ACM SIGGRAPH, pp. 279–288.

[34] Computer Graphics Lab, TU Braunschweig. Virtual Video Camera project website. `http://graphics.tu-bs.de/projects/vvc/`, Jan. 2013.

[35] Criminisi, A., Perez, P., and Toyama, K. Object removal by exemplar-based inpainting. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (2003), vol. 2, IEEE, pp. 721–728.

[36] Dana, K. J., van Ginneken, B., Nayar, S. K., and Koenderink, J. J. Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (Jan. 1999), 1–34.

[37] Davis, A., Levoy, M., and Durand, F. Unstructured light fields. *Comput. Graph. Forum 31*, 2 (2012), 305–314.

[38] de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S. Performance Capture from Sparse Multi-View Video. *ACM Trans. on Graphics 27*, 3 (2008), 1–10.

[39] Debevec, P. E., Taylor, C. J., and Malik, J. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 11–20.

[40] Deutscher Wetterdienst (DWD) - Abteilung Klima- und Umweltberatung. Tägliche Windspitzen in Beaufort von Deutschland 2011. `http://www.dwd.de`, Jan. 2012.

[41] Edelsbrunner, H., and Shah, N. Incremental topological flipping works for regular triangulations. *Algorithmica 15*, 3 (1996), 223–241.

[42] Einarsson, P., Chabert, C.-F., Jones, A., Ma, W.-C., Lamond, B., Hawkins, T., Bolas, M., Sylwan, S., and Debevec, P. Relighting Human Locomotion with Flowed Reflectance Fields. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering* (June 2006), pp. 183–194.

[43] Eisemann, M., Decker, B. D., Magnor, M., Bekaert, P., de Aguiar, E., Ahmed, N., Theobalt, C., and Sellent, A. Floating Textures. *Computer Graphics Forum (Proc. Eurographics EG'08) 27*, 2 (4 2008), 409–418.

[44] Emmerich, R. Independence Day (movie). `http://www.imdb.com/title/tt0116629/`, 1996.

[45] Enkelmann, W. Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. *Computer Vision, Graphics, and Image Processing 43*, 2 (1988), 150 – 177.

[46] Fehn, C. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. *Proc. SPIE 5291*, 93 (2004), 93–104.

[47] Felzenszwalb, P., and Huttenlocher, D. Efficient belief propagation for early vision. In *IJCV* (October 2006), vol. 70, pp. 41–54.

[48] Frey, B. J., and Dueck, D. Clustering by passing messages between data points. *Science 315* (2007), 972–976.

[49] Furukawa, Y., and Ponce, J. Dense 3d motion capture from synchronized video streams. In *CVPR* (2008), IEEE Computer Society, pp. 193–211.

[50] Furukawa, Y., and Ponce, J. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 8 (Aug. 2010), 1362–1376.

[51] Fusiello, A. Specifying virtual cameras in uncalibrated view synthesis. *Circuits and Systems for Video Technology, IEEE Transactions on 17*, 5 (may 2007), 604 –611.

[52] Fyffe, G., Hawkins, T., Watts, C., Ma, W.-C., and Debevec, P. Comprehensive facial performance capture. *Computer Graphics Forum 30*, 2 (2011), 425–434.

[53] Gal, R., Wexler, Y., Ofek, E., Hoppe, H., and Cohen-Or, D. Seamless montage for texturing models. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 479–486.

[54] Germann, M. *Video-based rendering techniques*. PhD thesis, Zürich, 2012. Diss., Eidgenssische Technische Hochschule ETH Zürich, Nr. 20290, 2012.

[55] Germann, M., Hornung, A., Keiser, R., Ziegler, R., Würmlin, S., and Gross, M. Articulated billboards for video-based rendering. *Comput. Graphics Forum (Proc. Eurographics) 29*, 2 (2010), 585–594.

[56] Germann, M., Popa, T., Keiser, R., Ziegler, R., and Gross, M. Novel-View Synthesis of Outdoor Sport Events Using an Adaptive View-Dependent Geometry. *Comput. Graphics Forum (Proc. Eurographics) 31*, 2 (2012), 325–333.

[57] Gherardi, R., Toldo, R., Farenzena, M., and Fusiello, A. Samantha: Towards automatic image-based model acquisition. In *Proceedings of the 2010 Conference on Visual Media Production* (Washington, DC, USA, 2010), CVMP '10, IEEE Computer Society, pp. 161–170.

[58] Glencross, M., Ward, G. J., Melendez, F., Jay, C., Liu, J., and Hubbold, R. A perceptually validated model for surface depth hallucination. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 59:1–59:8.

[59] Goesele, M., Ackermann, J., Fuhrmann, S., Haubold, C., Klowsky, R., and Darmstadt, T. Ambient point clouds for view interpolation. *ACM Trans. Graph. 29* (July 2010), 95:1–95:6.

[60] Goesele, M., Snavely, N., Curless, B., Hoppe, H., and Seitz, S. Multi-view stereo for community photo collections. *ICCV'07* (2007), 1–8.

[61] Goldlücke, B., and Magnor, M. Real-time microfacet billboarding for free-viewpoint video rendering. In *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on* (2003), vol. 3, IEEE, pp. 713–716.

[62] Goldlücke, B., and Magnor, M. Weighted minimal hypersurfaces and their applications in computer vision. 366–378.

[63] Goldlücke, B., and Magnor, M. Space-time isosurface evolution for temporally coherent 3d reconstruction. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (June-2 July), vol. 1, pp. I–350–I–355 Vol.1.

[64] google. Google Sketchup. `http://sketchup.google.com`, Nov. 2012.

[65] Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), SIGGRAPH '96, ACM, pp. 43–54.

[66] Grau, O. iview. `http://www.bbc.co.uk/rd/projects/iview/`, 2005.

[67] Guernsey, L. Turning the Super Bowl Into a Game of Pixels. `http://www.nytimes.com/2001/01/25/technology/news-watch-turning-the-super-bowl-into-a-game-of-pixels.html`, 2001.

[68] Guillemaut, J.-Y., Sarim, M., and Hilton, A. Stereoscopic content production of complex dynamic scenes using a wide-baseline monoscopic camera set-up. In *Proc. International Conference on Image Processing (ICIP 2010), Special Session on Image Processing for Stereo Digital Cinema Production* (2010), pp. 9–12.

[69] Hasler, N., Rosenhahn, B., Thormählen, T., Wand, M., Gall, J., and Seidel, H.-P. Markerless Motion Capture with Unsynchronized Moving Cameras. In *Proc. of CVPR'09* (Washington, June 2009), IEEE Computer Society, pp. 224–231.

[70] Hasler, N., Stoll, C., Sunkel, M., Rosenhahn, B., and Seidel, H.-P. A statistical model of human pose and body shape. *Comput. Graph. Forum 28*, 2 (2009), 337–346.

[71] Herbst, E., Seitz, S., and Baker, S. Occlusion Reasoning for Temporal Interpolation using Optical Flow. Tech. rep., Microsoft Research Technical Report, MSR-TR-2009-2014, 2009. no. MSR-TR-2009-2014.

[72] Hilton, A., Guillemaut, J.-Y., Kilner, J., Grau, O., and Thomas, G. 3d-tv production from conventional cameras for sports broadcast. *IEEE Transactions on Broadcasting 57*, 2 (2011), 462–476.

[73] Horn, B. K. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Tech. rep., Cambridge, MA, USA, 1970.

[74] Horn, B. K. P., and Schunck, B. G. Determining optical flow. *ARTIFICAL INTELLIGENCE 17* (1981), 185–203.

[75] Hornung, A., and Kobbelt, L. Interactive pixel-accurate free viewpoint rendering from images with silhouette aware sampling. *Computer Graphics Forum 28*, 8 (2009), 2090 – 2103.

[76] Hypr3D Development Team. hypr3D. `http://www.agisoft.ru/`, Nov. 2012.

[77] Imre, E., Guillemaut, J.-Y., and Hilton, A. Through-the-lens multi-camera synchronisation and frame-drop detection for 3d reconstruction. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on* (oct. 2012), pp. 395 –402.

[78] Imre, E., and Hilton, A. Through-the-lens synchronisation for heterogeneous camera networks. In *Proceedings of the British Machine Vision Conference* (2012), BMVA Press, pp. 97.1–97.11.

[79] Inc., D. A. Digital Air Techniques. http://www.digitalair.com/techniques/index.html, 2008.

[80] Ince, S., and Konrad, J. Occlusion-aware view interpolation. *EURASIP Journal on Image and Video Processing 2008* (2008), 1–15.

[81] Jain, A., Thormählen, T., Seidel, H.-P., and Theobalt, C. Moviereshape: Tracking and reshaping of humans in videos. *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2010) 29*, 5 (2010), 148:1–148:10.

[82] Kazhdan, M., Bolitho, M., and Hoppe, H. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2006), SGP '06, Eurographics Association, pp. 61–70.

[83] Klose, F., and Lipski, C. Making of "Symbiz - Who Cares" an Image-Based Stereoscopic Free Viewpoint Music Video. `http://www.fmx.de/about-us/history/fmx-2012/program-2012/event-details/event/390/611/51/611/detail/Event.html`, May 2012.

[84] Klose, F., Lipski, C., and Magnor, M. Reconstructing Shape and Motion from Asynchronous Cameras. In *15th International Workshop on Vision, Modeling and Visualization (VMV)* (Siegen, Germany, November 2010), pp. 171–177.

[85] Klose, F., Ruhl, K., Lipski, C., and Magnor, M. Flowlab - an interactive tool for editing dense image correspondences. In *Proceedings of the 2011 Conference for Visual Media Production* (Washington, DC, USA, 2011), CVMP '11, IEEE Computer Society, pp. 59–66.

[86] Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society 7*, 1 (1956), 48–50.

[87] Kuster, C., Popa, T., Zach, C., Gotsman, C., and Gross, M. Freecam: A hybrid camera system for interactive free-viewpoint video. In *Proceedings of Vision, Modeling, and Visualization (VMV)* (2011), pp. 17–24.

[88] Lee, S., Wolberg, G., and Shin, S. Polymorph: morphing among multiple images. *IEEE Computer Graphics and Applications* (1998), 58–71.

[89] Lempitsky, V., Boykov, Y., and Ivanov, D. Oriented visibility for multiview reconstruction. *Computer Vision–ECCV 2006* (2006), 226–238.

[90] Lempitsky, V. S., and Ivanov, D. V. Seamless mosaicing of image-based texture maps. In *CVPR* (2007), IEEE Computer Society, pp. 1–6.

[91] Lengyel, J. The convergence of graphics and vision. *Computer 31*, 7 (Jul), 46–53.

[92] Leutenegger, S., Chli, M., and Siegwart, R. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2548–2555.

[93] Levieux, P., Tompkin, J., and Kautz, J. Interactive viewpoint video textures. In *Proceedings of the 9th European Conference on Visual Media Production* (New York, NY, USA, 2012), CVMP '12, ACM, pp. 11–17.

[94] Levoy, M., and Hanrahan, P. Light Field Rendering. In *Proc. of ACM SIGGRAPH'96* (New York, 1996), ACM Press/ACM SIGGRAPH, pp. 31–42.

[95] Li, H., Luo, L., Vlasic, D., Peers, P., Popović, J., Pauly, M., and Rusinkiewicz, S. Temporally coherent completion of dynamic shapes. *ACM Trans. Graph. 31*, 1 (Feb. 2012), 2:1–2:11.

[96] Li, K., Dai, Q., Xu, W., and Yang, J. Temporal-dense dynamic 3-d reconstruction with low frame rate cameras. *Selected Topics in Signal Processing, IEEE Journal of 6*, 5 (2012), 447–459.

[97] LiberoVision AG. LiberoVision. `http://www.liberovision.com/index.php`, 2012.

[98] Linz, C. *Correspondence estimation and image interpolation for photo-realistic rendering.* PhD thesis, Braunschweig, 2011.

[99] Linz, C., Lipski, C., and Magnor, M. Multi-image interpolation based on graph-cuts and symmetric optic flow. In *Vision, Modeling, and Visualization (2010)* (2010), The Eurographics Association, pp. 115–122.

[100] Linz, C., Lipski, C., and Magnor, M. Multi-image interpolation based on graph-cuts and symmetric optical flow. In *ACM SIGGRAPH 2010 Posters* (2010), ACM, p. 129.

[101] Linz, C., Lipski, C., and Magnor, M. Multi-image interpolation based on graph-cuts and symmetric optical flow. In *Proc. Vision, Modeling and Visualization (VMV) 2010* (Siegen, Germany, Nov. 2010), Eurographics, Eurographics Association, pp. 115–122.

[102] Linz, C., Lipski, C., Rogge, L., Theobalt, C., and Magnor, M. Space-time visual effects as a post-production process. In *Proceedings of the 1st international workshop on 3D video processing* (New York, NY, USA, 2010), 3DVP '10, ACM, pp. 1–6.

[103] Lipski, C. Hands on HD 2011 - Virtual Video Camera. `http://www.nordmedia.de/bereichsfotos/serien/37971.608.html`, Aug. 2011.

[104] Lipski, C. Making of 3D-Gauss, Vision, Modeling, and Visualization Workshop 2009. `http://vmv09.tu-bs.de/index.php?page=gauss`, Nov. 2012.

[105] Lipski, C., Albuquerque, G., Stich, T., and Magnor, M. Spacetime Tetrahedra: Image-Based Viewpoint Navigation through Space and Time. Tech. rep., Computer Graphics Lab, TU Braunschweig, 2008. no. 2008-12-9.

[106] Lipski, C., Bose, D., Eisemann, M., Berger, K., and Magnor, M. Sparse bundle adjustment speedup strategies. In *WSCG Communication Papers Proceedings 2010* (Feb. 2010), pp. 85–88.

[107] Lipski, C., Klose, F., Ruhl, K., and Magnor, M. Making of Who Cares? HD Stereoscopic Free Viewpoint Video. In *Proc. European Conference on Visual Media Production (CVMP) 2011* (Nov. 2011), vol. 8, pp. 1–10.

[108] Lipski, C., Klose, F., Ruhl, K., and Magnor, M. The virtual video camera: Simplified 3DTV acquisition and processing. In *Proc. 3DTV-CON 2011* (Antalya, May 2011), IEEE Computer Society, pp. 1–4.

[109] Lipski, C., Linz, C., Berger, K., and Magnor, M. Virtual video camera: image-based viewpoint navigation through space and time. In *ACM SIGGRAPH 2009 Posters* (2009), ACM, p. 93.

[110] Lipski, C., Linz, C., Berger, K., Sellent, A., and Magnor, M. Virtual video camera: Image-based viewpoint navigation through space and time. *Computer Graphics Forum 29*, 8 (2010), 2555–2568.

[111] Lipski, C., Linz, C., and Magnor, M. Belief propagation optical flow for high-resolution image morphing. In *ACM SIGGRAPH 2010 Posters* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 67:1–67:1.

[112] Lipski, C., Linz, C., Neumann, T., and Magnor, M. High resolution image correspondences for video Post-Production. In *CVMP 2010* (London, 2010), pp. 1–8.

[113] Lipski, C., Linz, C., Neumann, T., Wacker, M., and Magnor, M. High resolution image correspondences for video post-production. *JVRB - Journal of Virtual Reality and Broadcasting*, 9 (2012), 1–10.

[114] Lipski, C., and Magnor, M. Stereo-3d-videoproduktion mit der virtual video camera. *FKT - Fachzeitschrift für Fernsehen, Film und elektronische Medien 2011*, 12 (Dec 2011), 692–695.

[115] Liu, C., Yuen, J., Torralba, A., Sivic, J., and Freeman, W. T. Sift flow: Dense correspondence across different scenes. In *ECCV '08: Proceedings of the 10th European Conference on Computer Vision* (2008), pp. 28–42.

[116] Liu, F., Gleicher, M., Jin, H., and Agarwala, A. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph. 28*, 3 (2009), 1–9.

[117] Liu, Y., Dai, Q., and Xu, W. A point-cloud-based multiview stereo algorithm for free-viewpoint video. *IEEE Transactions on Visualization and Computer Graphics 16*, 3 (May 2010), 407–418.

[118] Lourakis, M. I. A., and Argyros, A. A. Sba: A software package for generic sparse bundle adjustment. *ACM Trans. Math. Softw. 36*, 1 (2009).

[119] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (2004), 91–110.

[120] Lucas, B. D., and Kanade, T. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2* (San Francisco, CA, USA, 1981), IJCAI'81, Morgan Kaufmann Publishers Inc., pp. 674–679.

[121] Lytro, Inc. The Lytro camera. `https://www.lytro.com/`, Nov. 2012.

[122] Magnor, M., and Girod, B. Data compression for light-field rendering. *IEEE Trans. Cir. and Sys. for Video Technol. 10*, 3 (Apr. 2000), 338–343.

[123] Mahajan, D., Huang, F., Matusik, W., Ramamoorthi, R., and Belhumeur, P. Moving Gradients: A Path-Based Method for Plausible Image Interpolation. *ACM Trans. on Graphics* (2009), 600–608.

[124] Mark, W., McMillan, L., and Bishop, G. Post-Rendering 3D Warping. In *Proc. of Symposium on Interactive 3D Graphics* (1997), pp. 7–16.

[125] Matsuyama, T., Wu, X., Takai, T., and Nobuhara, S. Real-time 3D shape reconstruction, dynamic 3D mesh deformation, and high fidelity visualization for 3D video. *Computer Vision and Image Understanding 96*, 3 (2004), 393–434.

[126] Matusik, W., Buehler, C., Raskar, R., Gortler, S. J., and McMillan, L. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 369–374.

[127] McMillan, L., and Bishop, G. Plenoptic modeling: an image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 39–46.

[128] Meyer, B., Lipski, C., Scholz, B., and Magnor, M. Multi-view coding with dense correspondence fields. In *Proc. IEEE International Symposium on Consumer Electronics (ISCE) 2010* (June 2010), pp. 117–120.

[129] Meyer, B., Lipski, C., Scholz, B., and Magnor, M. Real-time Free-Viewpoint Navigation from Compressed Multi-Video Recordings. In *3DPVT '2010* (2010), pp. 1–6.

[130] Meyer, B., Stich, T., Magnor, M., and Pollefeys, M. Subframe Temporal Alignment of Non-Stationary Cameras. In *Proc. British Machine Vision Conference BMVC '08* (2008), pp. 11.1–11.10.

[131] Moeslund, T. B., Hilton, A., and Krüger, V. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst. 104*, 2 (Nov. 2006), 90–126.

[132] Montgomery, J. fxguidetv 138: CVMP Conference. http://www.fxguide.com/fxguidetv/cvmp2011/, Mar. 2012.

[133] Moreno-Noguer, F., Belhumeur, P. N., and Nayar, S. K. Active refocusing of images and videos. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM, pp. 67:1–67:9.

[134] Müller, K., Smolic, A., Dix, K., Kauff, P., and Wiegand, T. Reliability-based generation and view synthesis in layered depth video. In *Multimedia Signal Processing, 2008 IEEE 10th Workshop on* (2008), IEEE, pp. 34–39.

[135] Ncam. ncam - realtime camera tracking. http://www.ncam-tech.com, 2012.

[136] Nguyen, H., Wünsche, B., Delmas, P., and Lutteroth, C. 3d models from the black box: Investigating the current state of image-based modeling. *Journal of WSCG 20*, 1 (Feb. 2012), 1–10.

[137] Nocedal, J., and Wright, S. J. *Numerical Optimization*, 2nd ed. Springer, New York, 2006.

[138] Perona, P., and Malik, J. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 12*, 7 (1990), 629–639.

[139] Perry-Smith, L. Infinite-Realities. http://www.ir-ltd.net, 2012.

[140] Pixomondo. Hugo Cabret - Making of. http://vimeo.com/43914859, 2012.

[141] Potmesil, M. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing 40*, 1 (1987), 1 – 29.

[142] Pulli, K., Cohen, M., Duchamp, T., Hoppe, H., Shapiro, L., and Stuetzle, W. View-based rendering: Visualizing real objects from scanned range and color data. In *Eurographics Rendering Workshop 1997* (1997), pp. 23–34.

[143] Real Time Race. Real Time Race. http://www.realtimerace.com, 2011.

[144] Replay Technologies Inc. Replay technology showcases. http://www.replay-technologies.com/products/showcases, 2012.

[145] Rodriguez-Ramos, J., Marichal-Hernandez, J., Luke, J., Trujillo-Sevilla, J., Puga, M., Lopez, M., Fernandez-Valdivia, J., Dominguez-Conde, C., Sanluis, J., Rosa, F., Guadalupe, V., Quintero, H., Militello, C., Rodriguez-Ramos, L., Lopez, R., Montilla, I., and Femenia, B. New developments at cafadis plenoptic camera. In *Information Optics (WIO), 2011 10th Euro-American Workshop on* (june 2011), pp. 1 –3.

[146] Ruhl, K., Berger, K., Lipski, C., Klose, F., Schröder, Y., Scholz, A., and Magnor, M. Integrating multiple depth sensors into the virtual video camera. In *Proc. SIGGRAPH 2011* (Vancouver, Canada, Aug. 2011), ACM, ACM, p. 1. ACM SIGGRAPH 2011 Posters.

[147] Ruhl, K., Hell, B., Klose, F., Lipski, C., Petersen, S., and Magnor, M. Improving dense image correspondence estimation with interactive user guidance. In *Proceedings of the 20th ACM international conference on Multimedia* (New York, NY, USA, 2012), MM '12, ACM, pp. 1129–1132.

[148] Ruhl, K., Klose, F., Lipski, C., and Magnor, M. Integrating approximate depth data into dense image correspondence estimation. In *Proceedings of the 9th European Conference on Visual Media Production* (New York, NY, USA, 2012), CVMP '12, ACM, pp. 26–31.

[149] Saito, H., Baba, S., Kimura, M., Vedula, S., and Kanade, T. Appearance-based virtual view generation of temporally-varying events from multi-camera images in the 3d room. In *Proceedings of Second International Conference on 3-D Digital Imaging and Modeling* (October 1999), pp. 516 – 525.

[150] Sand, P., and Teller, S. Particle Video: Long-Range Motion Estimation using Point Trajectories. In *Proc. of CVPR'06* (Washington, 2006), IEEE Computer Society, pp. 2195–2202.

[151] Scharstein, D., and Szeliski, R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision 47*, 1-3 (Apr. 2002), 7–42.

[152] Scharstein, D., and Szeliski, R. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (2003), vol. 1, IEEE, pp. 195–202.

[153] Schmeing, M., and Jiang, X. Time-consistency of disocclusion filling algorithms in depth image based rendering. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2011* (2011), pp. 1–4.

[154] Schoenemann, T., and Cremers, D. High Resolution Motion Layer Decomposition using Dual-space Graph Cuts. In *Proc. of CVPR* (Washington, 2008), IEEE Computer Society, pp. 1–7.

[155] Schwartz, C., Ruiters, R., Weinmann, M., and Klein, R. Webgl-based streaming and presentation framework for bidirectional texture functions. In *The 12th International Symposium on Virtual Reality, Archeology and Cultural Heritage VAST 2011* (Oct. 2011), Eurographics Association, Eurographics Association, pp. 113–120.

[156] Schwartz, C., Schnabel, R., Degener, P., and Klein, R. Photopath: Single image path depictions from multiple photographs. *Journal of WSCG 18*, 1-3 (Feb. 2010).

[157] Schwartz, C., Weinmann, M., Ruiters, R., and Klein, R. Integrated high-quality acquisition of geometry and appearance for cultural heritage. In *The 12th International Symposium on Virtual Reality, Archeology and Cultural Heritage VAST 2011* (Oct. 2011), Eurographics Association, Eurographics Association, pp. 25–32.

[158] Seitz, S., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (2006), vol. 1, IEEE, pp. 519–528.

[159] Seitz, S. M., and Dyer, C. R. View Morphing. In *Proc. of ACM SIGGRAPH'96* (New York, 1996), ACM Press/ACM SIGGRAPH, pp. 21–30.

[160] Seymour, M. Art of Optical Flow. http://www.fxguide.com/featured/art_of_optical_flow/, 2006.

[161] Seymour, M. Heroic VFX : Stargate Digital and NBCs Heroes. http://www.fxguide.com/featured/heroic_vfx_stargate_digital_and_nbcs_heroes/, May 2007.

[162] Seymour, M. The Curious Case of Aging Visual Effects. http://www.fxguide.com/featured/the_curious_case_of_aging_visual_effects/, 2009.

[163] Seymour, M. Real Steel: case study in CGI / live action integration. http://www.fxguide.com/featured/the_curious_case_of_aging_visual_effects/, 2011.

[164] Seymour, M. Paul Debevec and ICT: an fxphd BKD. http://www.fxguide.com/featured/paul-debevec-and-ict-an-fxphd-bkd/, May 2012.

[165] Shade, J., Gortler, S., He, L., and Szeliski, R. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 231–242.

[166] Shi, J., and Tomasi, C. Good features to track. Tech. rep., Ithaca, NY, USA, 1993.

[167] Shum, H.-Y., and Kang, S. B. A review of image-based rendering techniques. *IEEE/SPIE Visual Communications and Image Processing (VCIP) 213* (2000).

[168] Si, H., and Gaertner, K. Meshing Piecewise Linear Complexes by Constrained Delaunay Tetrahedralizations. In *Proc. of International Meshing Roundtable* (Berlin, September 2005), Springer, pp. 147–163.

[169] Smith, B., Zhang, L., and Jin, H. Stereo matching with nonparametric smoothness priors in feature space. *IEEE Computer Vision and Pattern Recognition (CVPR) 0* (2009), 485–492.

[170] Smolic, A. An overview of 3d video and free viewpoint video. CAIP '09, Springer-Verlag, pp. 1–8.

[171] Snavely, N. Bundler: Structure from Motion (SfM) for Unordered Image Collections. http://phototour.cs.washington.edu/bundler/, Nov. 2012.

[172] Snavely, N., Garg, R., Seitz, S. M., and Szeliski, R. Finding paths through the world's photos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2008) 27*, 3 (2008), 11–21.

[173] Snavely, N., Seitz, S. M., and Szeliski, R. Photo tourism: Exploring photo collections in 3d. *ACM Trans. on Graphics 25*, 3 (2006), 835–846.

[174] Spielberg, S. Jurassic Park (movie). http://www.imdb.com/title/tt0107290/, 1993.

[175] Starck, J., and Hilton, A. Surface capture for performance-based animation. *Computer Graphics and Applications, IEEE 27*, 3 (may-june 2007), 21 –31.

[176] Stich, T. *Space-time interpolation techniques*. PhD thesis, Braunschweig, 2009.

[177] Stich, T., Linz, C., Albuquerque, G., and Magnor, M. View and Time Interpolation in Image Space. *Computer Graphics Forum (Proc. Pacific Graphics) 27*, 7 (2008), 1781–1787.

[178] Stich, T., Linz, C., Wallraven, C., Cunningham, D., and Magnor, M. Perception-motivated Interpolation of Image Sequences. In *Proc. of ACM APGV'08* (Los Angeles, USA, 2008), ACM Press, pp. 97–106.

[179] Stich, T., Linz, C., Wallraven, C., Cunningham, D., and Magnor, M. Perception-motivated interpolation of image sequences. *ACM Transactions on Applied Perception (TAP) 8*, 2 (Jan. 2011), 1–25. http://doi.acm.org/10.1145/1870076.1870079.

[180] Tauber, Z., Li, Z.-N., and Drew, M. Review and preview: Disocclusion by inpainting for image-based rendering. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 37*, 4 (2007), 527–540.

[181] The Foundry. Nuke/Cameratracker website. http://www.thefoundry.co.uk/products/, May 2012.

[182] The Foundry Visionmongers Ltd, BUF Compagnie, Trinity College Dublin, CERTH-ITI, Quantic Dream, University of Surrey. i3DPost - Final Annual Report, 2009-10. http://www.i3dpost.eu, 2010.

[183] The GIMP Team. GIMP - GNU Image Manipulation Program. http://www.gimp.org/, Jan. 2012.

[184] Time-Slice Films. Rip Curl Mirage Campaign. http://www.timeslicefilms.com/rip-curl-mirage, 2010.

[185] Time-Slice Films. Time-Slice Films. http://www.timeslicefilms.com, 2012.

[186] Tola, E., Lepetit, V., and Fua, P. Daisy: an Efficient Dense Descriptor Applied to Wide Baseline Stereo. vol. 32, pp. 815–830.

[187] Tomasi, C., and Kanade, T. Detection and tracking of point features. Tech. rep., International Journal of Computer Vision, 1991.

[188] Vaish, V., and Adams, A. The (New) Stanford Light Field Archive. http://lightfield.stanford.edu/, Nov. 2012.

[189] van den Hengel, A., Dick, A., Thormählen, T., Ward, B., and Torr, P. H. S. Videotrace: rapid interactive scene modelling from video. *ACM Trans. Graph. 26*, 3 (July 2007), 86:1–86:5.

[190] Vangorp, P., Chaurasia, G., Laffont, P.-Y., Fleming, R., and Drettakis, G. Perception of visual artifacts in image-based rendering of façades. *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering) 30*, 4 (2011), 1241–1250.

[191] Vedula, S., Baker, S., and Kanade, T. Image Based Spatio-Temporal Modeling and View Interpolation of Dynamic Events. *ACM Trans. on Graphics 24*, 2 (2005), 240–261.

[192] Vogiatzis, G., Hernandez, C., and Cipolla, R. Reconstruction in the round using photometric normals and silhouettes. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2* (Washington, DC, USA, 2006), CVPR '06, IEEE Computer Society, pp. 1847–1854.

[193] Wachowski, A., and Wachowski, L. The Matrix (movie). http://www.imdb.com/title/tt0133093/, 1999.

[194] Wang, H., Sun, M., and Yang, R. Space-time light field rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (July 2007), 697–710.

[195] Wang, L., Kang, S. B., Szeliski, R., and Shum, H.-Y. Optimal texture map reconstruction from multiple views. In *CVPR (1)*, IEEE Computer Society, pp. 347–354.

[196] Waschbüsch, M., Würmlin, S., and Gross, M. 3d video billboard clouds. In *Computer Graphics Forum* (2007), vol. 26, Wiley Online Library, pp. 561–569.

[197] Werlberger, M., Pock, T., and Bischof, H. Motion estimation with non-local total variation regularization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (San Francisco, CA, USA, June 2010), pp. 2464–2471.

[198] Wietzke, L. raytrix 3D lightfield camera technology. http://www.raytrix.de/, Nov. 2012.

[199] Wolf, M. Space, Time, Frame, Cinema: Exploring the Possibilities of Spatiotemporal Effects. *New Review of Film and Television Studies* (Dec. 2006), 369–374. www.digitalair.com/techniques/STFC.pdf.

[200] Woodham, R. Photometric method for determining surface orientation from multiple images. *Optical engineering 19*, 1 (1980), 139–144.

[201] Xú, G., and Zhang, Z. *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. Computational Imaging and Vision. Kluwer Academic Publishers, 1996.

[202] Xu, L., Chen, J., and Jia, J. A Segmentation Based Variational Model for Accurate Optical Flow Estimation. In *Proc. of ECCV'08* (Berlin, 2008), Springer, pp. 671–684.

[203] Yamazaki, S., Sagawa, R., Kawasaki, H., Ikeuchi, K., and Sakauchi, M. Microfacet billboarding. In *Proceedings of the 13th Eurographics workshop on*

Rendering (Aire-la-Ville, Switzerland, Switzerland, 2002), EGRW '02, Eurographics Association, pp. 169–180.

[204] Yang, R., Guinnip, D., and Wang, L. View-dependent textured splatting. *The Visual Computer 22*, 7 (2006), 456–467.

[205] Ye, M., Wang, X., Yang, R., Ren, L., and Pollefeys, M. Accurate 3d pose estimation from a single depth image. In *Proceedings of the 2011 International Conference on Computer Vision* (Washington, DC, USA, 2011), ICCV '11, IEEE Computer Society, pp. 731–738.

[206] Zabih, R., and Woodfill, J. Non-parametric local transforms for computing visual correspondence. In *Proceedings of the third European conference on Computer Vision (Vol. II)* (Secaucus, NJ, USA, 1994), ECCV '94, Springer-Verlag New York, Inc., pp. 151–158.

[207] Zhang, Z., Wang, L., Guo, B., and Shum, H.-Y. Feature-based light field morphing. *ACM Trans. on Graphics 21*, 3 (2002), 457–464.

[208] Zheng, K., Colburn, A., Agarwala, A., Agrawala, M., Salesin, D., Curless, B., and Cohen, M. Parallax photography: creating 3d cinematic effects from stills. In *Proceedings of Graphics Interface 2009* (2009), Canadian Information Processing Society, pp. 111–118.

[209] Zimmer, H., Bruhn, A., and Weickert, J. Freehand HDR imaging of moving scenes with simultaneous resolution enhancement. *Computer Graphics Forum (Proceedings of Eurographics) 30*, 2 (2011), 405–414. Web page: `http://www.mia.uni-saarland.de/Research/SR-HDR/`.

[210] Zitnick, C., Jojic, N., and Kang, S. B. Consistent Segmentation for Optical Flow Estimation. *ICCV'05 2* (2005), 1308–1315.

[211] Zitnick, C. L., Kang, S. B., Uyttendaele, M., Winder, S. A. J., and Szeliski, R. High-quality video view interpolation using a layered representation. *ACM Trans. Graph. 23*, 3 (2004), 600–608.

[212] Zubrzycki, J. Super Hi-Vision (SHV) for the London 2012 Olympics. `http://www.cvmp-conference.org/2012-Speakers/John-Zubrzycki`, 2012.

Author's publications

Basarke, C., Berger, K., Cornelsen, K., Doering, M., Effertz, J., Form, T., Gülke, T., Graefe, F., Hecker, P., Homeier, K., Klose, F., Lipski, C., Magnor, M., Morgenroth, J., Nothdurft, T., Ohl, S., Rauskolb, F. W., Rumpe, B., and Wolf, L. Team carolo - technical paper. Tech. Rep. 2008-7, Technische Universität Braunschweig, Carl-Friedrich-Gauss-Fakultät, Oct. 2008.

Berger, K., Linz, C., Lipski, C., Stich, T., and Magnor, M. Echtzeiterkennung von befahrbaren bereichen in urbanen szenarien. In *Proc. GI-Fachausschuß (FA) Echtzeitsysteme real-time 2008* (Oct. 2008), pp. 1–10.

Berger, K., Linz, C., Lipski, C., Vaudrey, T., Klette, R., and Magnor, M. Target space interactivity - the end of 3D widgets. In *WSCG Communication Papers Proceedings 2010* (Feb. 2010), pp. 1–8.

Berger, K., Lipski, C., Linz, C., Sellent, A., and Magnor, M. A ghosting artifact detector for interpolated image quality assessment. In *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization* (New York, NY, USA, 2009), APGV '09, ACM, pp. 128–128.

Berger, K., Lipski, C., Linz, C., Sellent, A., and Magnor, M. A ghosting artifact detector for interpolated image quality assessment. Tech. rep., TU Braunschweig, July 2009.

Berger, K., Lipski, C., Linz, C., Sellent, A., and Magnor, M. A ghosting artifact detector for interpolated image quality assessment. In *Consumer Electronics (ISCE), 2010 IEEE 14th International Symposium on* (2010), IEEE, pp. 1–6.

AUTHOR'S PUBLICATIONS

Berger, K., Lipski, C., Linz, C., Stich, T., and Magnor, M. The area processing unit of caroline-finding the way through darpas urban challenge. *Robot Vision* (2008), 260–274.

Klose, F., Linz, C., Lipski, C., and Magnor, M. Flexible stereoscopic 3D content creation of real world scenes. Tech. rep., Computer Graphics Lab, TU Braunschweig, Nov. 2010.

Klose, F., Lipski, C., and Magnor, M. Reconstructing Shape and Motion from Asynchronous Cameras. In *15th International Workshop on Vision, Modeling and Visualization (VMV)* (Siegen, Germany, November 2010), pp. 171–177.

Klose, F., Lipski, C., Ruhl, K., Meyer, B., and Magnor, M. A toolchain for capturing and rendering stereo and multi-view datasets. In *Proc. The International Conference on 3D Imaging (IC3D) 2011* (Dec. 2011), pp. 1–7.

Klose, F., Ruhl, K., Lipski, C., Linz, C., and Magnor, M. Stereoscopic 3D view synthesis from unsynchronized multi-view video. In *Proc. European Signal Processing Conference (EUSIPCO) 2011* (Barcelona, Spain, May 2011), F. Klose, K. Ruhl, C. Lipski, C. Linz, and M. Magnor, Eds., pp. 1904–1909.

Klose, F., Ruhl, K., Lipski, C., and Magnor, M. Flowlab - an interactive tool for editing dense image correspondences. In *Proceedings of the 2011 Conference for Visual Media Production* (Washington, DC, USA, 2011), CVMP '11, IEEE Computer Society, pp. 59–66.

Linz, C., Lipski, C., and Magnor, M. Multi-image interpolation based on graph-cuts and symmetric optic flow. In *Vision, Modeling, and Visualization (2010)* (2010), The Eurographics Association, pp. 115–122.

Linz, C., Lipski, C., and Magnor, M. Multi-image interpolation based on graph-cuts and symmetric optical flow. In *Proc. Vision, Modeling and Visualization (VMV) 2010* (Siegen, Germany, Nov. 2010), Eurographics, Eurographics Association, pp. 115–122.

Linz, C., Lipski, C., and Magnor, M. Multi-image interpolation based on graph-cuts and symmetric optical flow. In *ACM SIGGRAPH 2010 Posters* (2010), ACM, p. 129.

Linz, C., Lipski, C., Rogge, L., Theobalt, C., and Magnor, M. Space-time visual effects as a post-production process. In *Proceedings of the 1st international workshop on 3D video processing* (New York, NY, USA, 2010), 3DVP '10, ACM, pp. 1–6.

Lipski, C., Albuquerque, G., Stich, T., and Magnor, M. Spacetime Tetrahedra: Image-Based Viewpoint Navigation through Space and Time. Tech. rep., Computer Graphics Lab, TU Braunschweig, 2008. no. 2008-12-9.

Lipski, C., Berger, K., and Magnor, M. visage - A visualization and debugging framework for distributed system applications. In *Proc. WSCG 2009* (Plzen, Czech Republic, Feb. 2009), V. Skala, Ed., vol. 2009, UNION Agency – Science Press, pp. 1–7.

Lipski, C., Bose, D., Eisemann, M., Berger, K., and Magnor, M. Sparse bundle adjustment speedup strategies. In *WSCG Communication Papers Proceedings 2010* (Feb. 2010), pp. 85–88.

Lipski, C., Klose, F., Ruhl, K., and Magnor, M. Making of Who Cares? HD Stereoscopic Free Viewpoint Video. In *Proc. European Conference on Visual Media Production (CVMP) 2011* (Nov. 2011), vol. 8, pp. 1–10.

Lipski, C., Klose, F., Ruhl, K., and Magnor, M. The virtual video camera: Simplified 3DTV acquisition and processing. In *Proc. 3DTV-CON 2011* (Antalya, May 2011), IEEE Computer Society, pp. 1–4.

Lipski, C., Linz, C., Berger, K., and Magnor, M. Virtual video camera: image-based viewpoint navigation through space and time. In *ACM SIGGRAPH 2009 Posters* (2009), ACM, p. 93.

Lipski, C., Linz, C., Berger, K., Sellent, A., and Magnor, M. Virtual video camera: Image-based viewpoint navigation through space and time. *Computer Graphics Forum 29*, 8 (2010), 2555–2568.

Lipski, C., Linz, C., and Magnor, M. Belief propagation optical flow for high-resolution image morphing. In *ACM SIGGRAPH 2010 Posters* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 67:1–67:1.

AUTHOR'S PUBLICATIONS

Lipski, C., Linz, C., Neumann, T., and Magnor, M. High resolution image correspondences for video Post-Production. In *CVMP 2010* (London, 2010), pp. 1–8.

Lipski, C., Linz, C., Neumann, T., Wacker, M., and Magnor, M. High resolution image correspondences for video post-production. *JVRB - Journal of Virtual Reality and Broadcasting*, 9 (2012), 1–10.

Lipski, C., and Magnor, M. Stereo-3d-videoproduktion mit der virtual video camera. *FKT - Fachzeitschrift für Fernsehen, Film und elektronische Medien 2011*, 12 (Dec 2011), 692–695.

Lipski, C., Scholz, B., Berger, K., Linz, C., Stich, T., and Magnor, M. A fast and robust approach to lane marking detection and lane tracking. In *Proc. IEEE Southwest Symposium on Image Analysis and Interpretation 2008* (Washington, DC, USA, July 2008), vol. 2008, IEEE Computer Society, IEEE Computer Society, pp. 57–60.

Meyer, B., Lipski, C., Scholz, B., and Magnor, M. Multi-view coding with dense correspondence fields. In *Proc. IEEE International Symposium on Consumer Electronics (ISCE) 2010* (June 2010), pp. 117–120.

Meyer, B., Lipski, C., Scholz, B., and Magnor, M. Real-time Free-Viewpoint Navigation from Compressed Multi-Video Recordings. In *3DPVT '2010* (2010), pp. 1–6.

Rauskolb, F. W., Berger, K., Lipski, C., Magnor, M., Cornelsen, K., Effertz, J., Form, T., Graefe, F., Ohl, S., Schumacher, W., Wille, J. M., Hecker, P., Nothdurft, T., Doering, M., Homeier, K., Morgenroth, J., Wolf, L., Basarke, C., Berger, C., Gülke, T., Klose, F., and Rumpe, B. Caroline: An autonomously driving vehicle for urban environments. In *The DARPA Urban Challenge*, M. Buehler, K. Iagnemma, and S. Singh, Eds. Springer, Jan. 2010, pp. 441–508.

Rauskolb, F. W., Berger, K., Lipski, C., Magnor, M., Cornelsen, K., Effertz, J., Form, T., Graefe, F., Ohl, S., Schumacher, W., Wille, J. M., Hecker, P., Nothdurft, T., Doering, M., Homeier, K., Morgenroth, J., Wolf, L., Basarke, C., Berger, C., Gülke, T., Klose, F., and Rumpe, B.

Caroline: An autonomously driving vehicle for urban environments. *Journal of Field Robotics 25*, 9 (Dec. 2008), 674–724.

Rogge, L., Lipski, C., and Magnor, M. Visualization of the continental drift in real-time. *Journal of WSCG 18*, 1-3 (Mar. 2010), 9–16.

Ruhl, K., Berger, K., Lipski, C., Klose, F., Schröder, Y., Scholz, A., and Magnor, M. Integrating multiple depth sensors into the virtual video camera. In *Proc. SIGGRAPH 2011* (Vancouver, Canada, Aug. 2011), ACM, ACM, p. 1. ACM SIGGRAPH 2011 Posters.

Ruhl, K., Hell, B., Klose, F., Lipski, C., Petersen, S., and Magnor, M. Improving dense image correspondence estimation with interactive user guidance. In *Proceedings of the 20th ACM international conference on Multimedia* (New York, NY, USA, 2012), MM '12, ACM, pp. 1129–1132.

Ruhl, K., Klose, F., Lipski, C., and Magnor, M. Integrating approximate depth data into dense image correspondence estimation. In *Proceedings of the 9th European Conference on Visual Media Production* (New York, NY, USA, 2012), CVMP '12, ACM, pp. 26–31.

Salge, C., Lipski, C., Mahlmann, T., and Mathiak, B. Using genetically optimized artificial intelligence to improve gameplaying fun for strategical games. In *Proceedings of the 2008 ACM SIGGRAPH symposium on Video games 2008* (Los Angeles, California, Aug. 2008), vol. 2008, ACM, pp. 7–14.

AUTHOR'S PUBLICATIONS

Curriculum Vitæ - Lebenslauf

Curriculum Vitæ

1981	born in Braunschweig, Germany
2000	Highschool degree, main subjects English and chemistry
	Gymnasium Martino-Katharineum, Braunschweig, Germany
2001 - 2006	Diploma in Computer Science
	Technische Universität Braunschweig, Germany
since 2007	Ph.D. Student Computer Science, Prof. M. Magnor
	Computer Graphics Lab
	Technische Universität Braunschweig, Germany

Lebenslauf

1981	geboren in Braunschweig
2000	Allgemeine Hochschulreife
	Gymnasium Martino-Katharineum, Braunschweig, Germany
2001 - 2006	Diplom Informatik
	Technische Universität Braunschweig, Germany
seit 2007	Wissenschaftlicher Mitarbeiter, Prof. M. Magnor
	Institut für Computergraphik
	Technische Universität Braunschweig, Germany